GLOBAL OPTIMIZATION TECHNIQUES FOR NEURAL NETWORK APPLICATIONS

A Thesis submitted
in partial fulfilment of the requirement
for the degree of

MASTER OF TECHNOLOGY

by

A V M Manoj Kumar

To the

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

February 2002

CERTIFICATE

This is to certify that the work contained in this thesis entitled **"Global Optimization Techniques For Neural Network Applications"**, by A.V. M Manoj Kumar, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prem Kalra

Dr P K Kalra
Professor
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

ACKNOWLEDGEMENT

I'd like to take the opportunity to express my sincere thanks to my thesis supervisor Dr. P K Kalra for affording me an ambience where I could openly discuss my problems and ideas. It was a great pleasure to work under him for I gained much through the interactions.

My stay at IITK has been very fruitful for not only did I gain knowledge but also friends with whom I shared my views, met teachers who influenced my thought, and benign experiences that enriched my self and contributed to an all-round growth and well being. I thank my parents and my sister for all the support provided throughout, for their affections and encouragement.

I'd thank my batch-mates Saleem and Sankar for being with me all long as we did the courses together or shared a snack at Hall 4 canteen. My discussions with them gave me a feel for the algorithms I implemented for I could come up with certain schemes, which gave a direction of thought to the problems posed.

The atmosphere in the lab paved way for my betterment as I interacted with Madhav Krishna, Prashant, Maj. Jayesh, Maj. Vaid, Maj. Mohan Kumar, Maj. Patil, Manu and others. My sincere thanks to them. I am also thankful to Ashesh, Balaji and others who added all the flavors of joy and made my stay at IITK wonderful.

I would like to express my gratitude to all those who directly or indirectly helped me through the successful completion of my work.

A V M Manoj Kumar

ABSTRACT

Back propagation has often been applied to adapt artificial neural networks for various pattern classification and function approximation problems. However, an important limitation of this method is that it sometimes fails to find a global minimum of the error function. To avoid local minimum solutions in back propagation learning, Feedforward neural network training is treated as a global optimization problem.

Stochastic methods like Simulated Annealing and ALOPEX are investigated. Various cooling schedules for Simulated Annealing are developed. To improve the learning speed, hybrid algorithms are investigated. Hybrid-SA, SARPROP, NOVEL, R-GEM algorithms are developed. An improved variant of Hybrid-SA called SA-BP-SA is proposed. Various problems that hinder the learning process are dealt with suitable solutions. The algorithms developed are validated on several benchmark problems of functional approximation and classification.

CONTENTS

List of Figures

List of Tables

Terms used

SA	-	Simulated Annealing
ALOPEX	-	*Al*gorithm f*o*r Pattern *Ex*traction
Hybrid-SA	-	Hybrid-Simulated Annealing
SA-BP-SA	-	Simulated Annealing-Back Propagation-Simulated Annealing
SARPROP	-	Simulated Annealing Resilient Propagation
NOVEL	-	Non linear Optimization Via External Lead
R-GEM	-	Ray-Guided Exploration Method
GA	-	Genetic Algorithm
GA-SA	-	Genetic Algorithm – Simulated Annealing

CHAPTER 1

INTRODUCTION

Learning in neural networks is accomplished by adjusting the weights between connections in response to training patterns. In feed-forward neural networks, this involves mapping from an input space to an output space. That is, output O can be defined as a function of inputs I and connection weights $W: O = \Phi(I, W)$, where Φ represents a mapping function.

Learning involves finding a good mapping function – one that maps training patterns correctly and generalizes the mapping to test patterns not seen in training. This is done by adjusting weights W while fixing the topology and activation function. In other words, given a set of training patterns of input-output pairs $\{(I_1, D_1), (I_2, D_2), ..., (I_m, D_m)\}$ and an error function $\varepsilon(W, I_i, D_i)$, learning strives to minimize learning error $E(W)$:

$$\min_W E(W) = \min_W \sum_{i=1}^{m} \varepsilon(W, I_i, D_i) \qquad (1.1)$$

One popular error function is the average sum squared-error function in which $\varepsilon(W, I_i, D_i) = \frac{1}{2m} \sum_{i=1}^{m} (\Phi(I_i, W) - D_i)^2$. Since $E(W) \geq 0$ for a given set of training patterns, if a W^1 exists such that $E(W^1) = 0$, then W^1 is a global minimum; otherwise, the W that gives the smallest $E(W)$ is the global minimum. The quality of a learned network is measured by its error on a given set of training patterns and its (generalization) error on a given set of test patterns.

1.1 Problem Statement

Learning can be considered as an unconstrained nonlinear minimization problem in which the objective function is defined by error function and the search space is defined by weight space. Unfortunately, the terrain modeled by the error function in its weight space can be extremely rugged and have many local minima. This phenomenon is illustrated in Figure 1.1.

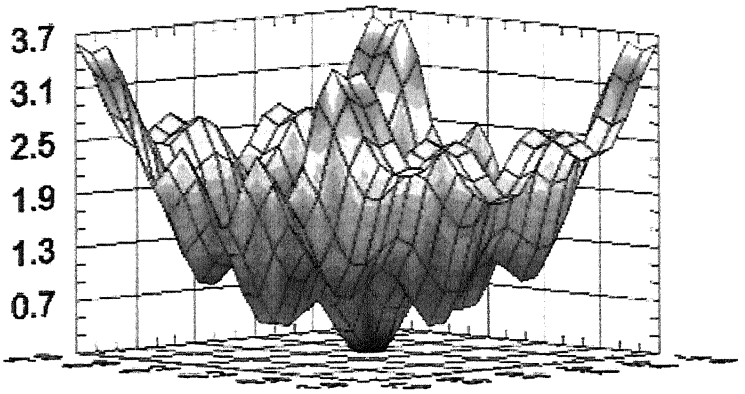

Fig 1.1 Error function with many Local Minima and a Global Minimum at (0,0)

Many learning algorithms find their roots in function-minimization algorithms that can be classified as local or global minimization algorithms. Back propagation is one of the local minimization algorithms used for training. It is simple and robust. It may not realize the global minima. In contrast, global-minimization algorithms have heuristic strategies to help escape from local minima. Our goal is improved learning of application problems that achieve less error-prone networks of the same size.

1.2 Error Minimization Methods

Learning feed-forward neural-network weights is like solving an unconstrained, continuous, non linear minimization problem. The task is to find variable assignments

that minimize the given objective function. The problem can be unimodal or multimodal, depending on the number of local minima in the objective function's space. In general, neural network learning is a multimodal, nonlinear Error minimization problem with many local minima. Generally, flat regions can mislead gradient-based methods. There can be many local minima that trap gradient based methods. Gradients may differ by many orders of magnitude, making it difficult to use gradients in any search method. A good search method should, therefore, have mechanisms to use gradient information to perform local search and escape from a local minimum after getting there.

Local minimization algorithms, such as gradient descent and Newton's method, find local minima efficiently and work best in unimodal problems. Global minimization methods, in contrast, employ heuristic strategies to look for global minima and do not stop after finding a local minimum.

Local-minimization algorithms have difficulties when the surface is flat (gradients close to zero), when gradients can be in a large range, or when the surface is very rugged. When gradients can vary greatly, the search may progress too slowly when the gradient is small and may overshoot when the gradient is large. When the error surface is rugged, a local search from a random starting point usually converges to a local minimum close to the initial point and a worse solution than the global minimum. Moreover, these algorithms require choosing some parameters and incorrectly chosen parameters can cause slow convergence.

To overcome local-search deficiencies, global-minimization methods have been developed. Figure1.2 classifies unconstrained, nonlinear, Error minimization methods. These methods can be classified as probabilistic or deterministic. They use local search

to determine local minima and focus on bringing the search out of a local minima once it gets there.

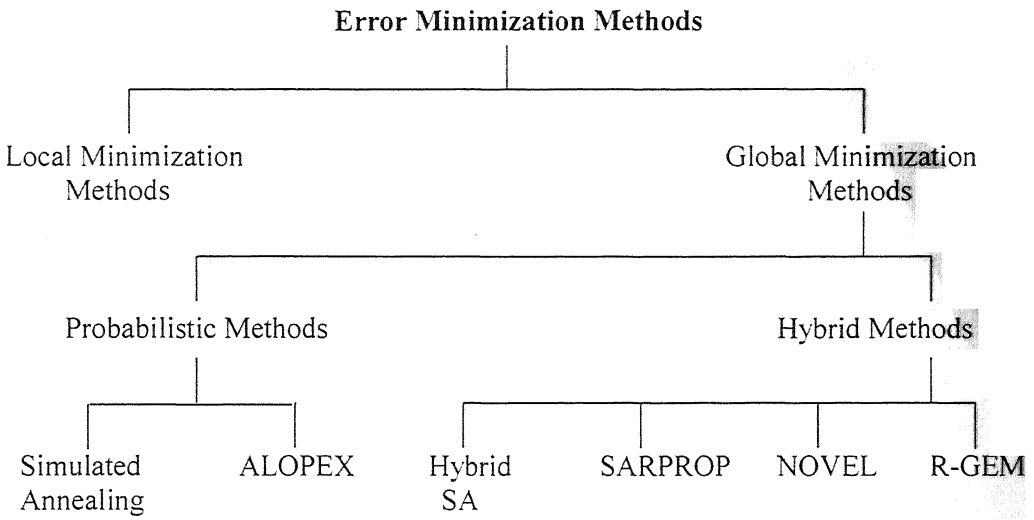

Figure 1.2 Classification of unconstrained, nonlinear, continuous Error-Minimization Methods.

Probabilistic global-minimization methods rely on probability to make decisions. The simplest probabilistic algorithm uses restarts to bring a search out of a local minimum when little improvement can be made locally. More advanced methods rely on probability to indicate whether a search should ascend from a local minimum – for example, simulated annealing when it accepts uphill movements.

Though stochastic methods like Simulated Annealing, ALOPEX overcomes local minima and realizes global minimum; it takes a long time for convergence. To improve the speed of the algorithm, hybrid methods are investigated. Hybrid methods typically combine an efficient local method able to find the local minima and a global exploration method able to escape from local minima and ensuring the convergence toward the global minimum.

1.3 Organization of Thesis

The objective of the thesis is to study the techniques of overcoming local minima in neural networks. Chapter 2 deals with the probabilistic global optimization algorithms. Chapter 3 deals with the hybrid global optimization algorithms. The algorithms developed are validated on various benchmark problems in Chapter 4. The conclusions drawn and future scope are addressed in Chapter 5.

CHAPTER 2

STOCHASTIC METHODS

Stochastic methods are non-deterministic in nature. These methods do not use gradient information. Thus these are applicable to a wider range of error functions, specifically, functions whose gradients are expensive to compute or functions that are not differentiable. Simulated Annealing and ALOPEX are two such type methods.

2.1 Simulated Annealing

Simulated Annealing is analogous to the physical behavior of annealing a molten metal. Above its melting temperature, a metal enters a phase where atoms (particles) are positioned at random according to statistical mechanics. As with all physical systems, the particles of the molten metal seek minimum energy configurations (states) if allowed to cool. A minimum energy configuration means a highly ordered state such as a defect-free crystal lattice. In order to achieve the defect-free crystal, the metal is annealed: First, the metal is heated above its melting point and then cooled slowly until it solidifies into a "perfect" crystalline structure. Slow cooling is necessary to prevent dislocations of atoms and other crystal lattice disruption. The defect-free crystal state corresponds to the global minimum energy configuration. A brief presentation of related statistical mechanics concepts that helps in understanding the underlying principles of the simulated annealing global optimization method is given in Appendix-A.

Two operations are involved in simulated annealing: a thermostatic operation that schedules decreases in the temperature and a stochastic relaxation that iteratively finds the

equilibrium solution at the new temperature using the final state of the system at the previous temperature as a starting point. Simulated annealing introduces artificial thermal noise that is gradually decreased over time. This noise is controlled by temperature. Noise allows occasional hill climbing interspersed with descents. The idea is to apply uniform random perturbations to the weight vector(ΔW) to the search point(W) and then to determine the resulting change in error function(E). If the value of the error function is reduced (i.e. $\Delta E < 0$), the new search point $(W' = W + \Delta W)$ is accepted. On the other hand, if the perturbation leads to an increase in E (i.e., $\Delta E > 0$), the new search point W' may or may not be accepted. In this case, the determination of whether to accept W' is stochastic, with probability $e^{-(\Delta E/T)}$. Hence, for large values of T, the probability of an uphill move in E is large. However, for small T, the probability of an uphill move is low; i.e., as T decreases, fewer uphill moves are allowed. This leads to an effective guidance of search, since the uphill moves are done in a controlled fashion, and thus there is no danger of jumping out of local minimum and falling into a worse one.

2.1.1 Algorithm

The objective is to minimize an error measure, E, with respect to network weights W, for a given set of training samples. The algorithm can be described as follows:

Step 1.

Create the network. Initialize the weights randomly. Compute the Network errorsum by forward-passing the training patterns.

Step 2.

Compute $W' = W + \Delta W$,

where ΔW is a small uniform random perturbation.

Compute the network error by forward-passing the patterns.

Compute the change in errorsum ΔE.

if ($\Delta E < 0$)

Accept the weight change

else

Compute $exp(-\Delta E / T)$

Generate a random number 'r' between 0 and 1

if ($exp\ (-\Delta E / T) < r$)

Accept the weight change.

else

Reject the weight change.

Step 3.

Increment *epoch* by 1.

Increment *counter* by 1.

Step 4. *if* (*counter* = *countermax*)

Update T according to the annealing schedule.

Reset *counter* to zero.

Step 5. Repeat steps 2 to 4 until network error approaches a desired value or the epoch criterion is met - which ever is earlier.

2.1.2 Cooling Schedules

The effectiveness of simulated annealing in locating global minima and its speed of convergence critically depends on the choice of cooling schedule for T. If the cooling schedule is too fast, a premature convergence to a local minimum might occur, and if it is too slow, the algorithm will require an excessive amount of computation time to converge. A discussion of a number of different cooling schedules can be found in the literature. The following are the cooling schedules that are investigated in this work.

2.1.2.1 Geometric cooling schedule

Geometric cooling is described by the temperature-update formula: $T_{k+1} = \alpha T_k$.

The cooling factor, α, is assumed to be a constant and less than one. This scheme is probably the most commonly used in the Simulated Annealing Literature, and acts as a base line for comparison with other more elaborate schemes.

2.1.2.2 Multiple cooling rates

The simple geometric cooling schedule always applies the same cooling rate regardless of the state of the system. In physical systems, it is known that a substance may undergo a number of phase transitions during cooling, before it reaches its frozen state. At each phase transition the physical structure of the substance undergoes a change which alters its behavior. At high temperatures the gross structure of the solution is resolved, and at low temperatures the minute details of the solution are resolved.

The geometric cooling scheme applies the same cooling factor regardless of the phase. However, at high temperatures almost all proposed interchanges are accepted, even though many of them could be non-productive. The application of a different cooling rate, that depends on the phase, would allow the algorithm to spend less time in the high

temperature phases. Consequently, more time would be spent in the low temperature phases, thus reducing the total amount of time required to solve the problem. At the same time, the high temperature phase cannot be omitted because it is responsible for setting up the gross structure of the solution.

In order to compute the temperature at which a phase transition occurs it is necessary to calculate the specific heat of the substance. A phase transition occurs when the specific heat is maximal, and this triggers the change in the state ordering. It is possible to compute the specific heat by observing the temperature at which the expression $\frac{\sigma^2(E)}{T}$ is maximal. In practice, we need to calculate the variance of the error function over a number of trials at a particular temperature, T. the background theory of derivations are available in van Larrhoven & Aarts [10].

The new cooling schedule is defined below. The temperature is reduced geometrically in this schedule. However, two different cooling rates, α and β, are applied depending on whether the system is above or below the temperature at which the specific heat is maximal:

$T_{k+1} = \alpha T_k$ if $T_k > T_{msp}$

$T_{k+1} = \beta T_k$ if $T_k \leq T_{msp}$

where T_{msp} is the temperature at which the maximum specific heat occurs

α and β are constants <1, and $\alpha < \beta$

As in the simple geometric cooling scheme, typical values of α might be 0.5 for a fast cooling rate, and then values of 0.9 or 0.99 for β, yielding much slower cooling after the transition.

2.1.2.3 Power cooling schedule

Power cooling is described by the temperature-update formula: $T_{k+1} = (T_k)^{\alpha}$. The cooling factor, α, is assumed be a constant and less than one. Typical value of α is 0.99.

2.1.2.4 Exponential cooling schedule

Exponential cooling is described by the temperature-update formula: $T_{k+1} = T_0 e^{-\alpha * t}$. The cooling factor, t, is nothing but the number of epoch during training. α is assumed be a constant and less than one. Typical value of α is 0.01.

2.1.2.5 Logarithmic cooling schedule

Logarithmic cooling is described by the temperature-update formula: $T_k = \frac{T_0}{1 + \ln k}$.

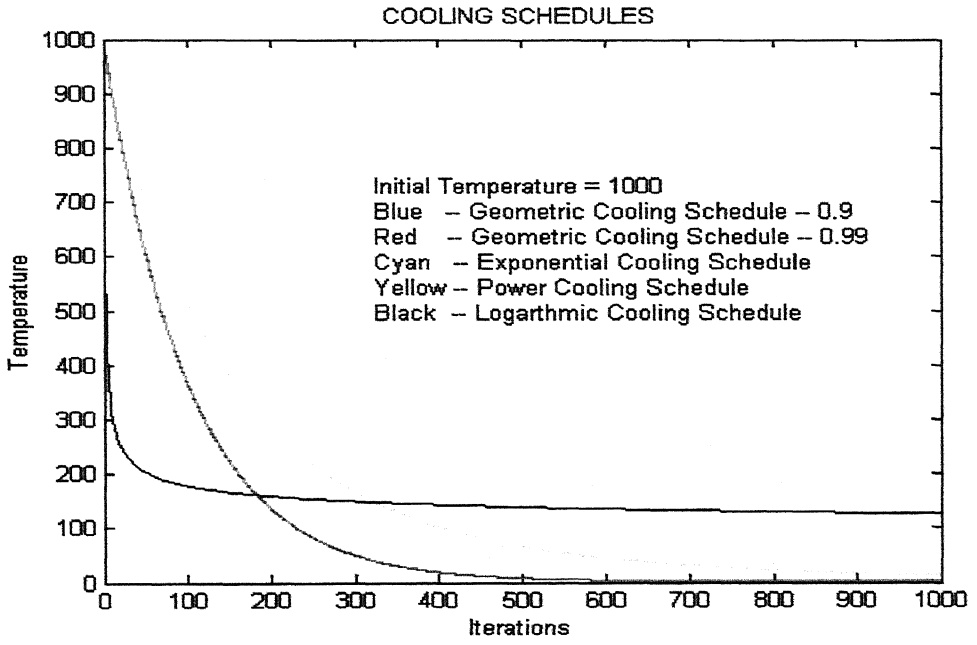

Fig 2.1 Different Cooling schedules

2.2 ALOPEX

ALOPEX is an acronym for *Al*gorithm for *P*attern *ex*traction. The algorithm can be described as follows: consider a neuron i with an interconnection strength w_{ij} from neuron j. During the nth iteration, the weight w_{ij} is updated according to the rule,

$$w_{ij}(n) = w_{ij}(n-1) + \delta_{ij}(n) \tag{2.1}$$

where $\delta_{ij}(n)$ is a small positive or negative step of size δ with the following probabilities:

$$\begin{aligned} \delta_{ij}(n) &= -\delta \quad \text{with probability } p_{ij}(n) \\ &= +\delta \quad \text{with probability } 1 - p_{ij}(n) \end{aligned} \tag{2.2}$$

The probability $p_{ij}(n)$ for a negative step is given by the Boltzmann distribution:

$$p_{ij}(n) = \frac{1}{1 + e^{-\frac{C_{ij}(n)}{T(n)}}} \tag{2.3}$$

where $C_{ij}(n)$ is given by the correlation:

$$C_{ij}(n) = \Delta w_{ij}(n) * \Delta E(n) \tag{2.4}$$

and $T(n)$ is a positive 'temperature'. $\Delta w_{ij}(n)$ and $\Delta E(n)$ are the changes in weight w_{ij} and the error measure E over the previous two iterations

$$\begin{aligned} \Delta w_{ij}(n) &= w_{ij}(n-1) - w_{ij}(n-2) \\ \Delta E(n) &= E(n-1) - E(n-2) \end{aligned} \tag{2.5}$$

(For the first two iterations, weights are chosen randomly)

The 'temperature' T in equation (2.3) is updated every N iterations using the following annealing schedule:

$$T(n) = \frac{\delta}{N} \sum_{n'=n-N}^{n-1} \left| \Delta E(n') \right| \tag{2.6}$$

2.2.1 Behavior of the Algorithm

From equations (2.1) – (2.3), if ΔE is negative, the probability of moving each weight in the same direction is greater than 0.5. If ΔE is positive, the probability of moving each weight in the opposite direction is greater than 0.5. In other words, the algorithm favors weight changes that will decrease the error E.

The temperature in equation (2.3) determines the stochasticity of the algorithm. With a non-zero value for T, the algorithm takes biased random walks in the weight space towards decreasing E. If T is too large, the probabilities are close to 0.5 and the algorithm does not settle into the global minimum of E. If T is too small, it gets trapped in local minima of E. Hence the value of T for each iteration is chosen very carefully. This can be done by choosing a heuristic annealing schedule. Update T at regular intervals to the average absolute value of the correlation C_{ij} over that interval. This method automatically reduces T when the correlations are small and increases T in regions of large correlations. The correlations need to be averaged over sufficiently large number of iterations so that the annealing does not freeze the algorithm at local minima. Towards the end, the step size δ can also be reduced for precise convergence. The use of a controllable temperature and the use of probabilistic parameter updates are similar to the method of simulated annealing. Typical values of N are 10 to 100, δ are 0.0001 to 0.01.

2.3 Advantages

1. These algorithms don't require any gradient information for updating weights.

2. When there is no a priori information available to choose a search direction, these perform minimization of Error function.

3. When E is a complex function with a large number of weight variables and local minima, these will converge eventually to the global minimum.

2.4 Disadvantages

1. These algorithms are usually slower than conventional back-propagation algorithms.

2. As the Error function becomes smaller, the convergence becomes even slower.

CHAPTER 3

HYBRID METHODS

Though stochastic methods overcome local minima and realize global minimum; it takes a long time to converge. To improve the speed of the algorithm, hybrid methods are investigated. Hybrid methods typically combine an efficient local method able to find the local minima and a global exploration method able to escape from local minima and ensuring the convergence toward the global minimum. Hybrid-SA, SARPROP, NOVEL and R-GEM are the hybrid methods investigated in this chapter.

3.1 Description of Hybrid-SA Method

This method makes use of both the merits of the back propagation method and simulated annealing method to find the global minimum of the error function of a neural network in a small number of steps. In this method, the back-propagation method is utilized to find a local minimum of $E(W)$. The Simulated Annealing method is used to ensure convergence to the global minimum of $E(W)$.

3.1.1 Hybrid-SA Algorithm

The outline of the hybrid algorithm is described as in figure 3.1. First, parameter training is carried out using the back-propagation method. When the decrease of the value of the error function $E(W)$ becomes smaller than the specified value ε^1, the algorithm shifts from back propagation method to simulated annealing method to prevent it from falling into a local minimum of $E(W)$. If the decrease of the error function becomes larger than $\max(E(W^k)G, \varepsilon)(0 < G < 1)$, the algorithm shifts from simulated annealing to the back

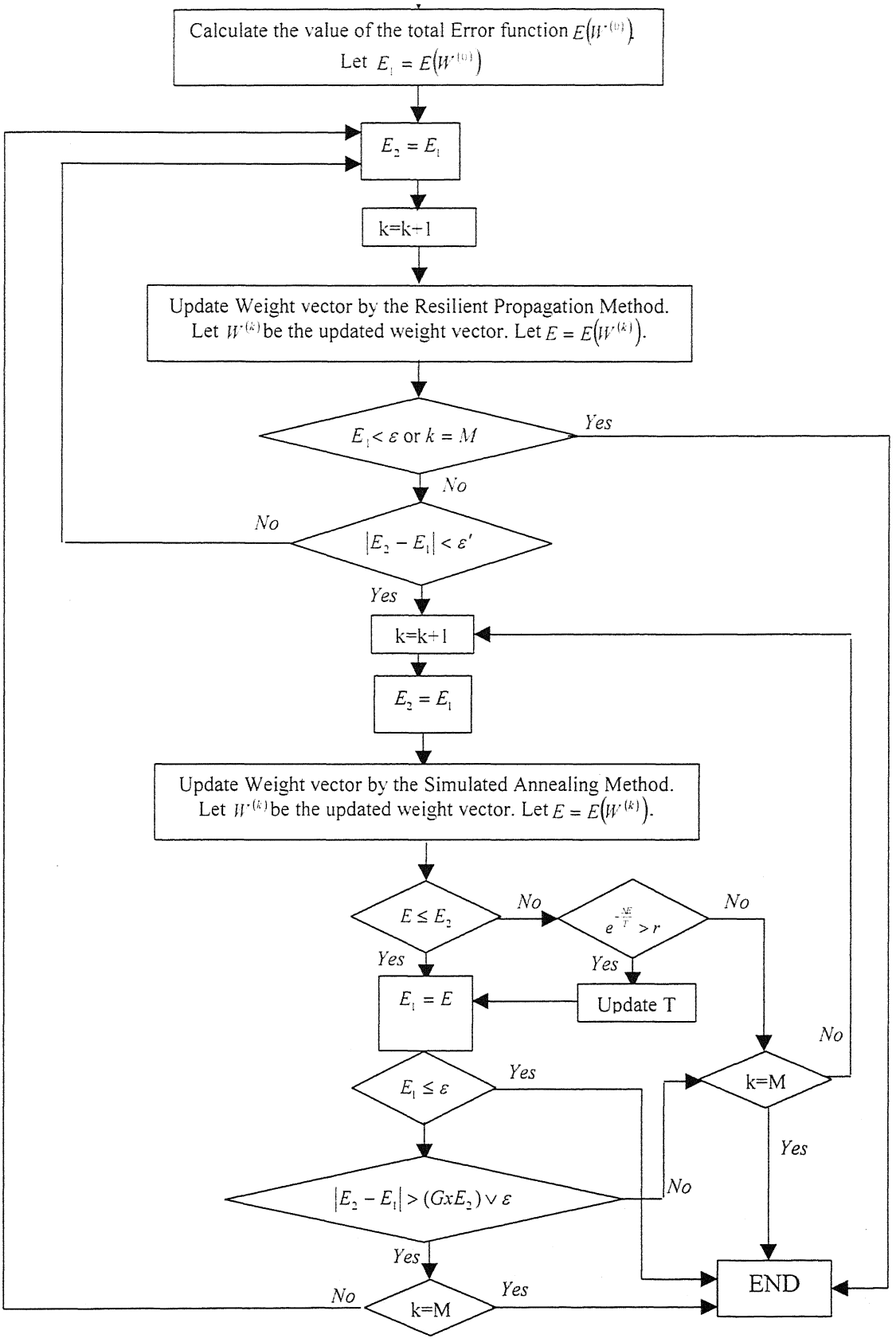

Figure 3.1 Hybrid Algorithm of RPROP and Simulated Annealing.

propagation. The fact that decrease of the error function has become larger than $\max(E(W^k)G,\varepsilon)$ means that the current weight vector $W^{(k+1)}$ has jumped from a local minimum point into a new valley of the error function $E(W)$. This change is performed to speed up the overall minimization. The same procedures are repeated several times. The basic idea of the algorithm is shown in figure 3.2. When the total number of steps exceeds a specified number, the overall calculation is stopped.

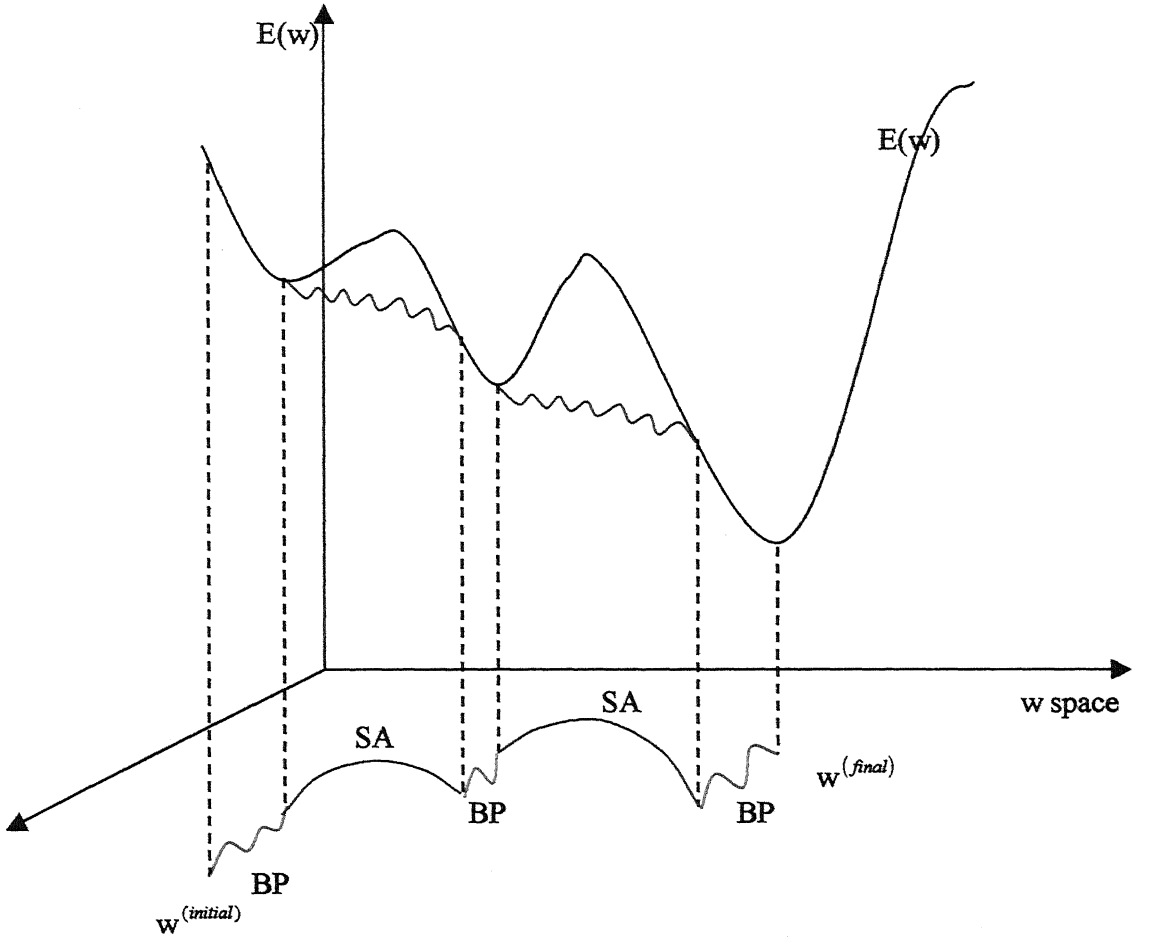

Fig 3.2 Basic idea of the Hybrid-SA algorithm

Resilient back propagation (RPROP) is used instead of back propagation in the implementation of the algorithm. RPROP algorithm is given in Appendix-2. It has performed very well on benchmark problems. The condition of changing control from

RPROP to Simulated Annealing is detection of excessive fluctuations in error surface or detection of flat error function. The condition of changing control from Simulated Annealing to RPROP is same as described in figure 3.1.

3.1.2 SA-BP-SA

In the conventional Hybrid-SA method, the weights are initialized randomly. Where as in SA-BP-SA, the initialized weights are tuned properly by Simulated Annealing. Simulated Annealing is performed on the network for certain number of epochs (varies from problem to problem) and then Hybrid-SA routine is performed. It is found that SA-BP-SA method gives fast and consistent results than Hybrid-SA method. The termination criterion from SA to Hybrid-SA is epoch-based one. One other criterion is reduction in error by some fixed percentage or when there is no significant reduction in error sum.

3.2 Description of SARPROP

The acronym for SARPROP is *S*imulated *A*nnealing *R*esilient *Prop*agation. While RPROP can be extremely fast in converging to a solution, it can often converge to poor local minima. SARPROP attempts to address this problem by using the method of Simulated Annealing (SA). SA in general, involves the addition of noise to the parameters undergoing optimization. The amount of noise added is associated with a SA term, which decreases the effect of the noise as training progresses. The addition of noise allows the network to move in a direction, which is not necessarily the direction of steepest descent. The benefit this provides is that it can help the network escape from local minima.

In SARPROP, noise is added to the standard RPROP weight update value when both the error gradient changes sign in successive epochs, and the magnitude of the update value is less than a value proportional to the SA term. The amount of noise added is proportional

to the SA term. The reason for adding noise to the update value only when both the error gradient changes sign, and the update value is below a given setting, is to minimize the disturbance to the normal adaptation of the update value. Following this scheme means that the update value is only modified by noise when it has a relatively small value (indicating a number of previous gradient crossings). This can allow the weight to jump out of local minima, while minimizing the disturbance to the adaptation process.

SARPROP uses SA not only on the noise added to the weight updates, but also on the amount of weight decay the network uses. Weight decay is implemented in SARPROP by adding a penalty term to the error function, which results in a modification of the error gradient. The weight decay term in SARPROP is designed such that small valued weights decay more rapidly than larger weights. This decay term allows important functionality already learned by the network (and represented by the large weights) to be retained more easily.

The SA term applied to the weight decay in SARPROP results in the influence of the weight decay decreasing as training proceeds. The SARPROP error gradient is shown below

$$\partial E/\partial w_{ij}^{SARPROP} = \partial E/\partial w_{ij} - 0.01 * w_{ij} / \left(1 + w_{ij}^2\right) * SA \qquad (3.1)$$

where $SA = 2^{-T*epoch}$ and T = temperature

The effect of this form of weight decay is to modify the error surface so that initially smaller weights are favored. As training progresses the weight decay magnitude is reduced, which modifies the error surface to allow for the easier growth of large weights. The modification of the error surface allows the network to explore regions of the error surface which were previously unavailable. The use of weight decay is a form of regularization, which has been found to improve generalization.

The SARPROP algorithm is as follows:

$$\forall i,j : \Delta_{ij}(t) = \Delta_0$$

$$\forall i,j : \frac{\partial E}{\partial w_{ij}(t-1)} = 0$$

REPEAT

Compute SARPROP Gradient $\frac{\partial E}{\partial w_{ij}(t)}$

For all weights and biases

IF $\frac{\partial E}{\partial w_{ij}(t-1)} * \frac{\partial E}{\partial w_{ij}(t)} > 0$

$$\Delta_{ij}(t) = \textbf{Minimum}\left(\Delta_{ij}(t-1) * \eta^{+}, \Delta_{\max}\right)$$

$$\Delta w_{ij}(t) = \textbf{-sign}\left(\frac{\partial E}{\partial w_{ij}(t)}\right) * \Delta_{ij}(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

$$\frac{\partial E}{\partial w_{ij}(t-1)} = \frac{\partial E}{\partial w_{ij}(t)}$$

ELSEIF $\frac{\partial E}{\partial w_{ij}(t-1)} * \frac{\partial E}{\partial w_{ij}(t)} < 0$

IF $\left(\Delta_{ij}(t-1) < 0.4 * SA^2\right)$

$$\Delta_{ij}(t) = \Delta_{ij}(t-1) * \eta^{-} + 0{,}8 * r * SA^2$$

ELSE

$$\Delta_{ij}(t) = \Delta_{ij}(t-1) * \eta^{-}$$

$$\Delta_{ij}(t) = \textbf{Maximum}\left(\Delta_{ij}, \Delta_{\min}\right)$$

$$\frac{\partial E}{\partial w_{ij}(t-1)} = 0$$

ELSEIF $\frac{\partial E}{\partial w_{ij}(t-1)} * \frac{\partial E}{\partial w_{ij}(t)} = 0$

$$\Delta w_{ij}(t) = \textbf{-sign}\left(\frac{\partial E}{\partial w_{ij}(t)}\right) * \Delta_{ij}(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

$$\frac{\partial E}{\partial w_{ij}(t-1)} = \frac{\partial E}{\partial w_{ij}(t)}$$

UNTIL CONVERGED

In the SARPROP algorithm, r is a random number between zero and one. The only parameter requiring setting prior to training is the temperature T, a part of the SA term, which affects the speed by which the noise and weight decay are reduced. The optimal setting of this parameter is problem dependent. In general it was found the more complex the problem, the lower the temperature value required (and hence the slower the annealing process). With these enhancements to RPROP, the likelihood of escaping from the local minima is increased; there is still no guarantee. The RPROP/SARPROP algorithms sometimes result in oscillations in the error values. These situations are shown in Figures 3.3 and 3.4. The following section addresses this situation.

3.2.1 Modifications to SARPROP (ReSARPROP)

Oscillations occur in the error values because of termination of SA mode. After considerable iterations (parameter dependent), the SA term that is added to the gradient term has negligible effect.

The modifications done to the algorithm are as follows. When the error value oscillates, the current weight values are used as the new initial weights, so that any prior training knowledge is maintained. The SA terms are reset, which may allow the network to jump out from its current local minimum and perhaps converge to a better solution. This algorithm will be termed ReSARPROP.

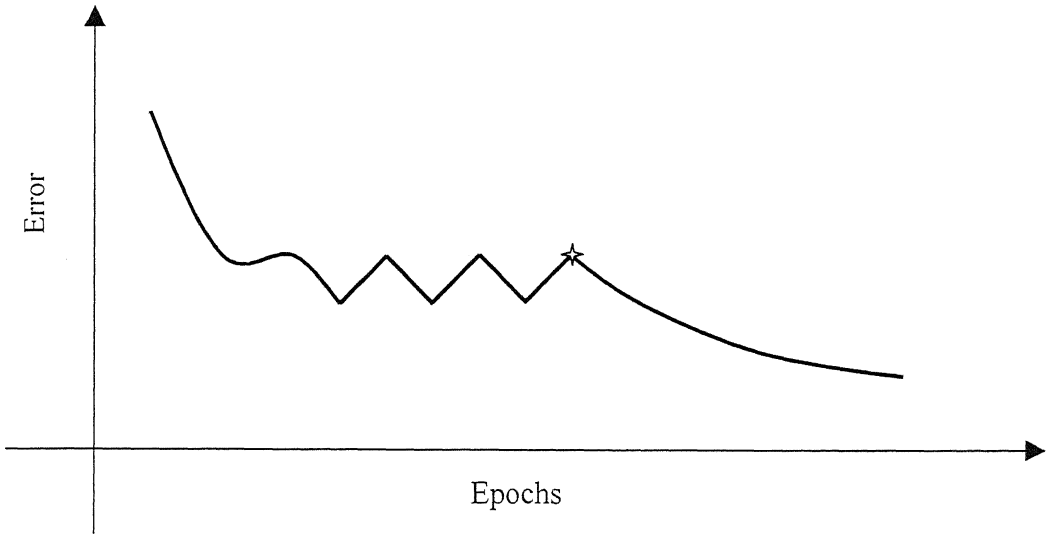

Fig 3.3 SAPROP Oscillations Case 1

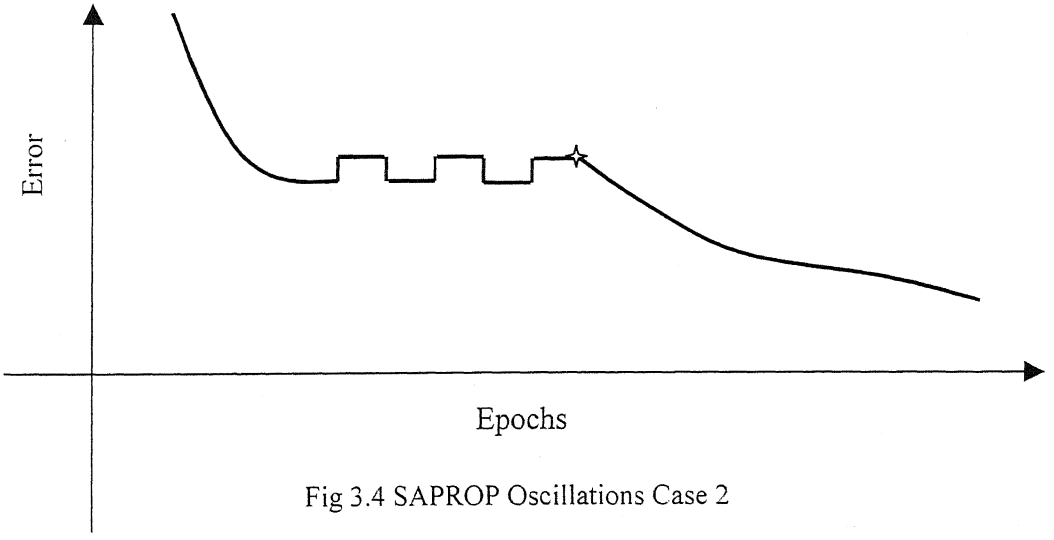

Fig 3.4 SAPROP Oscillations Case 2

The ReSARPROP algorithm is as follows:

Training is started by SARPROP

Keep track of error values. Store the latest six error values.

Check for the following conditions

```
IF (E [6]> E [5])
    IF (E [5]< E [4])
        IF (E [4]> E [3])
            IF (E [3]< E [2])
                IF (E [2]> E [1])
OR

  IF (E [6]= = E [5])
    IF (E [5]< E [4])
        IF (E [4]= = E [3])
            IF (E [3]> E [2])
                IF (E [2]= = E [1])


THEN
        Reset the SA parameters.
```

Continue the training process.

SARPROP requires the Temperature parameter 'T' to be set prior to training, and the optimal value depends on the problem. Using ReSARPROP, the temperature can be initially set to give fast annealing. If a good solution has not been reached, this temperature can be reset to allow for slower annealing when the network is reset. The removal of the temperature parameter in ReSARPROP results in ReSARPROP being (user) parameter free. The training is started by SARPROP with the default value of 'T', and as ReSARPROP is applied, the value of 'T' is reset based on an exponential decrease.

3.3 NOVEL

The acronym for NOVEL is *N*onlinear *O*ptimization *v*ia *E*xternal *L*ead. It is one of the powerful methods for finding global minima. This has a deterministic mechanism to bring a search out of a local minimum. A trajectory-based method, NOVEL relies on an external force to pull the search out of a local minimum and employs local descents to locate local minima.

It has three features: exploring the solution space, locating promising regions and finding local minima. In exploring the solution space, the search is guided by a continuous terrain-independent trace that does not get trapped in local minima. In locating promising regions, NOVEL uses a local gradient to attract the search to a local minimum, but it relies on the trace to pull out of the local minimum once little improvement can be found. Finally, NOVEL selects one initial point for each promising local region; it uses these initial points for a descent algorithm to find local minima.

NOVEL is efficient in that it tries to identify good starting points before applying a local search. This identification avoids searching unpromising local minima from random starting points using computationally expensive descent algorithms.

3.3.1 NOVEL framework

NOVEL has two phases: global search to identify regions containing local minima and local search to actually find them.

The global-search phase has a number of bootstrapping stages (Figure 3.5 shows three). The dynamics in each stage is represented by an ordinary differential equation. The first-stage input trace function is predefined. Then, each stage couples to the next stage by feeding its output trajectory as the next-stage trace function.

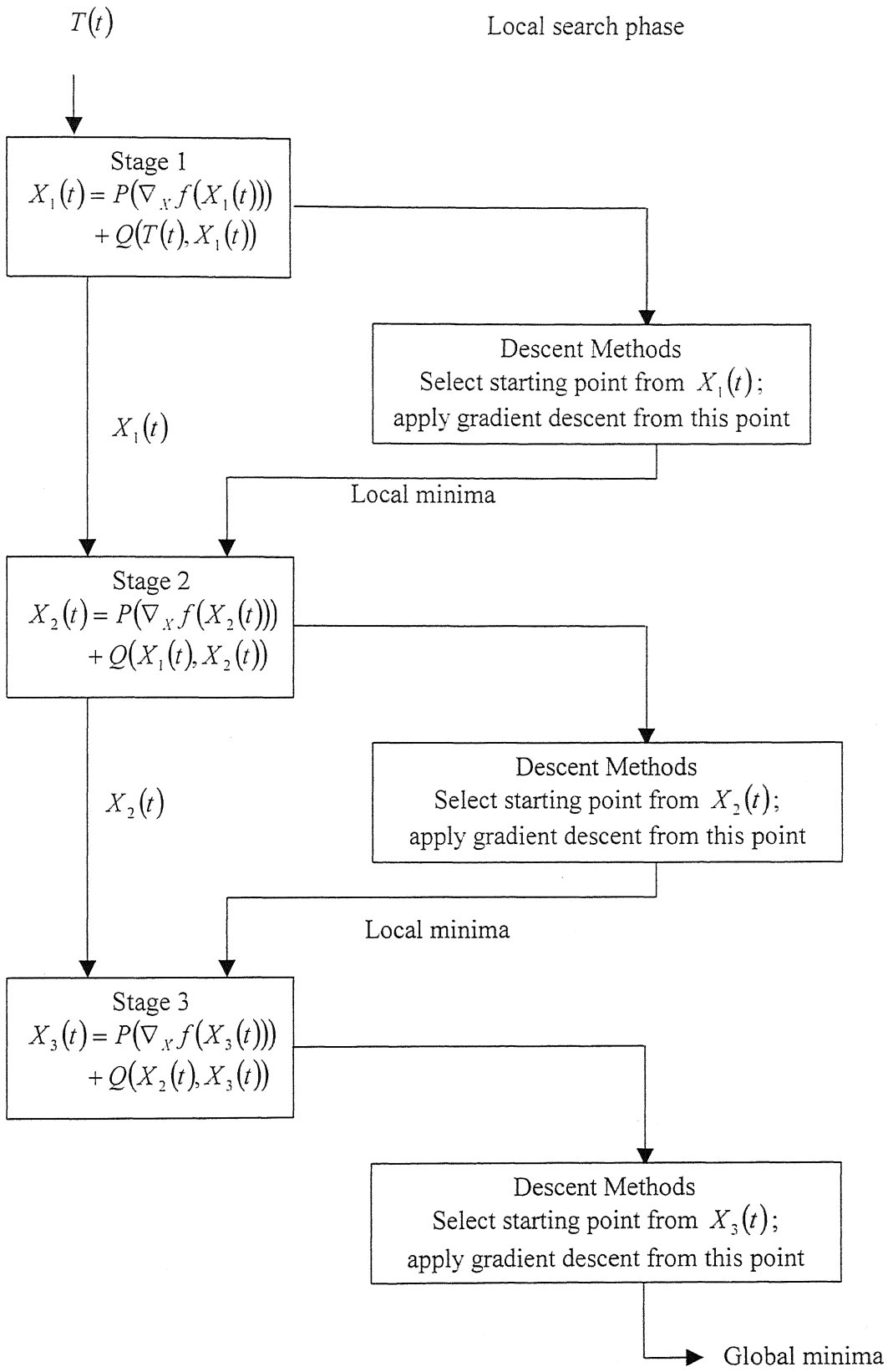

Fig 3.5 Framework of NOVEL method (3-Stage Global Search)

In general, the equations in each stage can be different. In earlier stages, more weight can be placed on the trace function, allowing the resulting trajectory to explore more regions. In later stages, more weight can be placed on local descents, allowing the trajectory to descend deeper into local basins. All equations in the global-search phase could be combined into a single equation before being solved, but that could limit each trajectory's identification of new starting points leading to better local minima.

In the local-search phase, we apply a traditional descent method, such as back propagation method or resilient propagation method to find local minima. The selection of initial points is based on trajectories output by the global-search phase.

3.3.2 Global-search phase

Assume $f(X)$ with gradient $\nabla_X f(X)$ is to be minimized, where $X = (x_1, x_2, ..., x_n)$ are variables. Each stage in the NOVEL global-search phase defines a trajectory $X(t) = (x_1(t), ..., x_n(t))$ that is governed by the following ordinary differential equation:

$$\dot{X}(t) = P(\nabla_X f(X(t))) + Q(T(t), X(t)) \tag{3.2}$$

where t is the autonomous variable; T, the trace function, is a function of t; and P and Q are general nonlinear functions. This equation specifies a trajectory through variable space X. It has two components: $P(\nabla_X f(X))$ lets the gradient attract the trajectory to a local minimum, and $Q(T, X)$ lets the trace function lead the trajectory out of the local minimum.

P and Q can have various forms. We used a simple form

$$\dot{X}(t) = -\mu_g \nabla_X f(X(t)) - \mu_t (X(t) - T(t)) \tag{3.3}$$

where μ_g and μ_t are constant coefficients.

Finding the global minima, without terrain knowledge, requires a trace function that covers the search space uniformly. There are two alternatives: Divide the space into subspaces and search one extensively before going to another, or search the space from coarse to fine. Numerous dimensions can make the first approach impractical, so we chose the second approach. The trace function is given by

$$T_i(t) = \rho \sin\left[2\pi \left(\frac{t}{2} \right)^{1-(0.05+0.45(i-1)/n)} + \frac{2\pi(i-1)}{n} \right] \tag{3.3}$$

where i represents the ith dimension, ρ is a coefficient specifying the range, and n is the number of dimensions.

We used a difference-equation solver for solving equation 4. The difference equation derived is

$$X(t+\delta t) = X(t) + \delta t \left[-\mu_g \nabla_X f(X(t)) - \mu_t (X(t) - T(t)) \right] \tag{3.4}$$

where δt is the step size. A large δt causes a large stride of variable modification, possibly resulting in oscillations. On the other hand, a small δt requires a longer computation time to traverse the same distance. This approach is fast and allows both patternwise learning and epochwise learning.

3.4 Ray-guided Global Optimization Method

This method expands the region of exploration and enables the Back propagation algorithm to achieve better performance. The algorithm starts from a local minimum and uses searching rays around the weight vector corresponding to the minimum to collect terrain information. During the search, promising starting points are identified, from which BP is applied to find the minima.

3.4.1 Ray-guided exploration method(R-GEM)

The Ray-guided exploration method (R-GEM) combines global search to identify promising starting points and local search to find the minima corresponding to the starting points. The algorithm first finds one minimum $W(0)$, from which global search guided by emanating rays is launched. The emanating rays can be written as

$$W^{(m)}(t) = W^{(m)}(0) + T^{(m)}(t) \tag{3.5}$$

Where $W^{(m)} = \left(w_1^{(m)}, w_2^{(m)}, \ldots\ldots, w_n^{(m)}\right), w_n^{(m)} \in R$ is the connection weight vector, m is the search ray index, t is the ray parameter and $T^{(m)}(t)$ indicates the direction of the ray.

The emanating ray plays an important role in uncovering new regions with potential local minima. Each ray is parameterized by the autonomous variable t. At t=0, the ray is at the starting minimum. As t increases, it moves away from the starting minimum. The directions of the rays are randomly chosen. The random emanating rays are governed by the following equation:

$$T^{(m)}(t) = t * D^{(m)} \tag{3.6}$$

Where $D^{(m)} = \left(d_1^{(m)}, d_2^{(m)}, \ldots\ldots, d_n^{(m)}\right), d_n^{(m)} \in R$ is a normalized vector randomly generated. As rays advance with t, the starting points are obtained by utilizing the magnitude of error

along the emanating rays. The identification of the starting points is accomplished by searching for dips along the ray.

Each time a starting point is identified; BP is applied to find the corresponding local minimum. The procedure is repeated until some stopping conditions are satisfied. After searching along the specified number of rays, the minimum and its corresponding weight vector with the best performance is found.

3.4.2 Description of the Algorithm

The R-GEM algorithm is described by the following steps.

Step 1: Apply BP to find a minimum $W^{(0)}(0)$.

Step 2: Randomly generate a searching ray with direction $D^{(m)} = \left(d_1^{(m)}, d_2^{(m)}, \ldots\ldots, d_n^{(m)}\right)$.

Step 3: Search along the ray to find the dips in error level.

Step 4: Apply BP to each of the dips found in step 3. If the minimum offers better performance, then update using the corresponding vector.

Step 5: Repeat steps 2-4 for a specified number of times.

R-GEM tries to identify good starting points before applying local search. Hence it avoids searching unpromising regions from random starting points. Since each of the searching rays of R-GEM is terrain-independent, the rays are not interrelated to each other. The calculations of the searching rays are therefore independent of each other

In applying the R-GEM to the training of the network, we start from a point generated randomly in the range of [-1,1] in each dimension. The range of t decides the range of global search. eq.(1) is used for discretised the computer simulation.

$$W^{(m)}(t+\Delta t) = W^{(m)}(t) + \Delta t * D^{(m)} \qquad (3.7)$$

where Δt is the step size. Too large a Δt may cause loss of information. On the other hand too small a Δt requires excessive computation. It is found that $\Delta t \in (0.005, 0.02)$ gives good results for the classification problems in our problems.

CHAPTER 4

RESULTS AND DISCUSSION

In the present work, some of the benchmark problems that were used by researchers to validate their algorithms have been used to validate the developed algorithms. The problems include Exclusive-OR, 4-Bit parity, and Two Spiral problems, which come into the category of classification problems. For function approximation problems, Sin(x)*Sin(y) and a three Input Nonlinear Function (TINF) have been considered. For prediction problems, Mackey-Glass Time Series Prediction problem has been considered. The simulation results of these problems on various algorithms discussed earlier are given in the following sections.

4.1 Results by Simulated Annealing (SA)

4.1.1 The Exclusive-OR Problem

This is a two input-one output problem. Output is zero when both the inputs are equal (0 or 1), and output is one when they are unequal. This problem is presented to Simulated Annealing. The architecture used is 2-2-1. The specified error tolerance is 1.0E-5. The corresponding error plots during training are shown in fig 4.1.1.1. Table 4.1 provides brief overview of the results where different cooling schedules are applied.

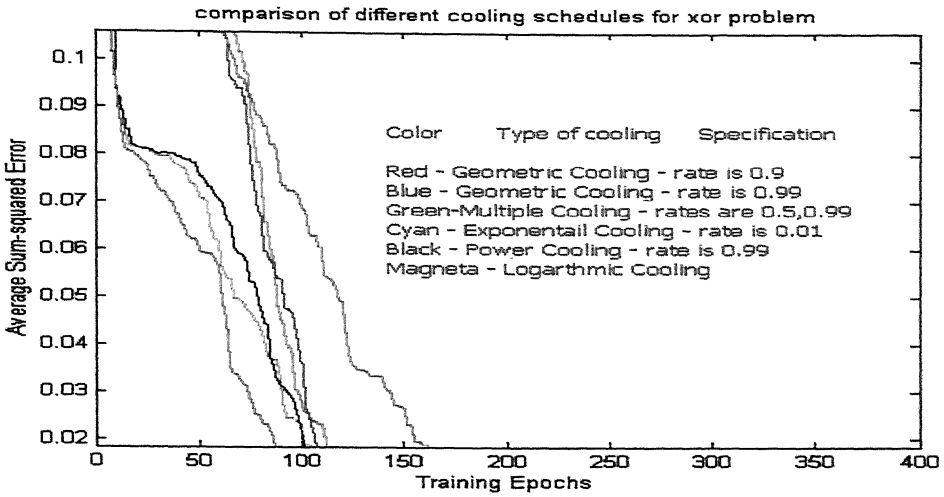

Fig 4.1.1.1 XOR by SA

Cooling Schedule	Cooling factor	Network Architecture	Approximate Iterations
Geometric	0.9	2-2-1	995
Geometric	0.99	2-2-1	429
Multiple	0.5,0.9	2-2-1	1581
Power	0.99	2-2-1	370
Exponential	0.01	2-2-1	400
Logarithmic	-	2-2-1	2000

Table 4.1 Comparison of cooling schedules for XOR problem

We observed that logarithmic schedule results are inferior compared to that of others because the annealing process is too slow in this schedule. Power and Exponential schedules have given good results.

4.1.2 Sin(x)*Sin(y) Problem

General function approximation problems have been used by different researchers to validate their network capabilities and learning algorithms etc. A popular function used for approximation is the Sin(x)*Sin(y). From the grid points of the range $[0,2\pi]\times[0,2\pi]$ within the input space of the above equation, 121 training data pairs were obtained. 76 testing data pairs were obtained from the intervals.

The network used to solve this problem is 2-6-5-6-1. It takes more epochs to reach global minimum. The reason being increase in weight space (In this case, the number of weights is 96). The inputs are normalized in the range of 0.1 to 0.9. The results are tabulated in 4.2 and the plots are shown in fig 4.1.2. The convergence reached by the network is 1e-5. This clearly indicates that SA is computationally expensive when the dimensionality is more than 20 variables (Here variables means weights).

Cooling Schedule	Cooling factor	Network Architecture	Approximate Iterations
Geometric	0.9	2-6-5-6-1	50000
Geometric	0.99	2-6-5-6-1	50000
Multiple	0.5,0.9	2-6-5-6-1	55000
Power	0.99	2-6-5-6-1	40000
Exponential	0.01	2-6-5-6-1	40000
Logarithmic	-	2-6-5-6-1	45000

Table 4.2 Comparison of different cooling schedules - Sin(x)*Sin(y) - SA

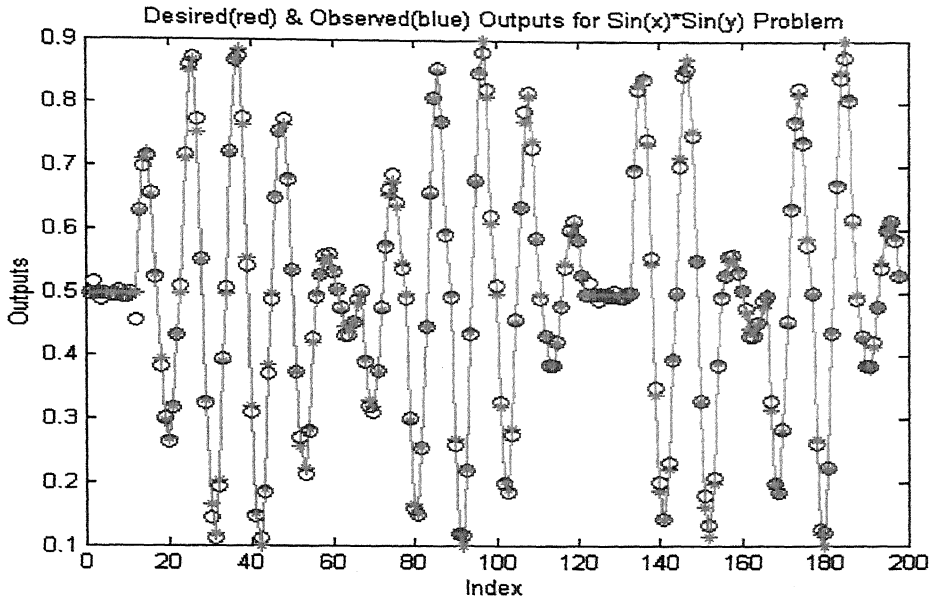

Fig 4.1.2.1 Desired & Observed Outputs – Sin(x)*Sin(y) – SA

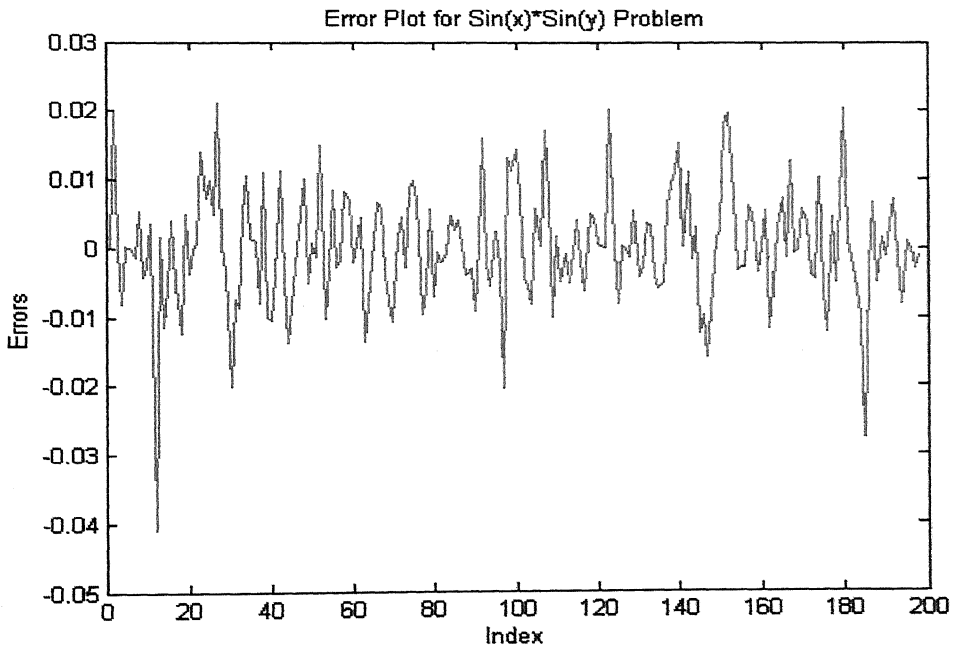

Fig 4.1.2.2 Errors at each pattern – Sin(x)*Sin(y) – SA

4.1.3 Predicting Chaotic Dynamics

The time series used in this simulation is generated by the chaotic Mackey-Glass differential delay equation defined below,

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \qquad (6.1)$$

The prediction of future values of this time series is a benchmark problem, which has been considered by a number of researchers. The goal of the task is to use known values up to the point x=t to predict the value at some point in the future x=t+P. The standard method for this type of prediction is to create a mapping from D points of the time series spaced *d* apart, that is, (x(t-(D-1)*d*),...,x(t-*d*),x(t)), to a predicted future value x(t+P). The value D=4 has been used in this simulation.

From the Mackey-Glass time series x(t), 1000 input-output data pairs have been extracted, of which the first 500 have been used for training and the rest for testing. The simulation results are shown in Table 4.1.3.1

Cooling Schedule	Cooling factor	Network Architecture	Approximate Iterations
Geometric	0.9	4-8-1	10000
Multiple	0.5,0.99	2-6-5-6-1	95000
Power	0.99	2-6-5-6-1	12000
Exponential	0.01	2-6-5-6-1	16000

Table 4.3 Comparison of different cooling schedules – Mackey Glass Time Series Prediction - SA

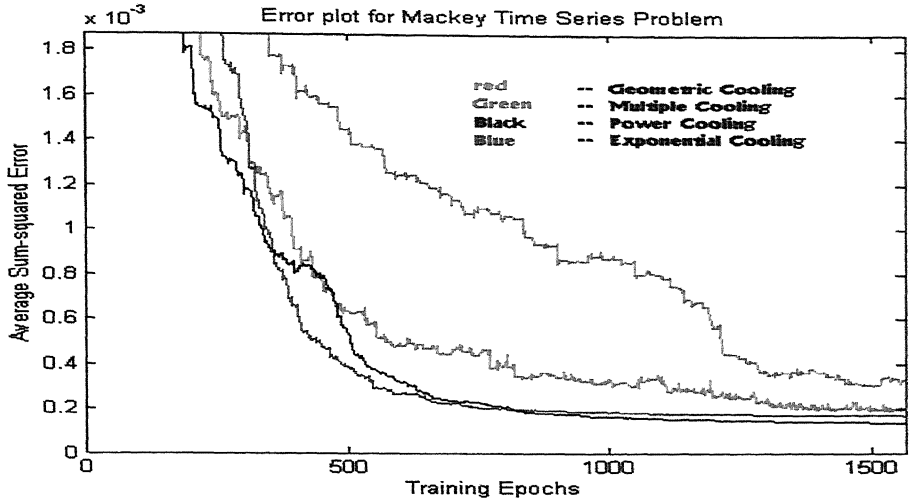

Fig 4.1.3.1 Comparison of various cooling schedules – Mackey-Glass Time Series Prediction Problem

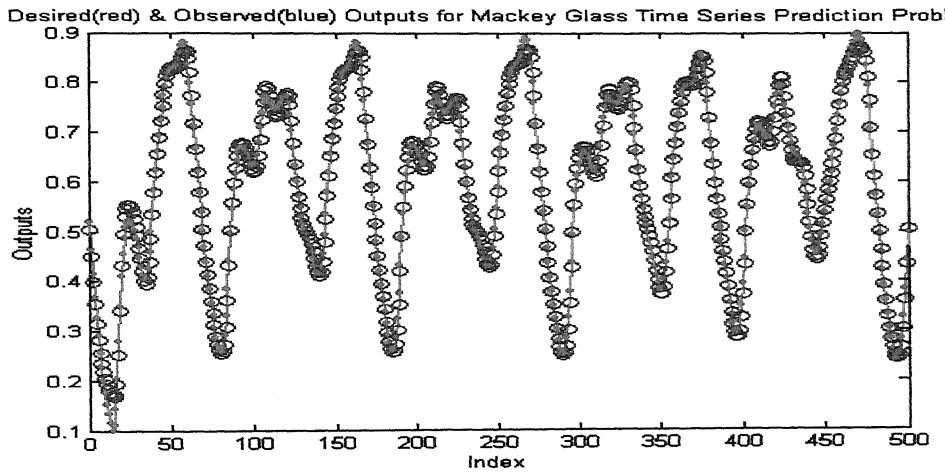

Fig 4.1.3.2 Mackey-Glass Time Series Prediction -Test Results – SA

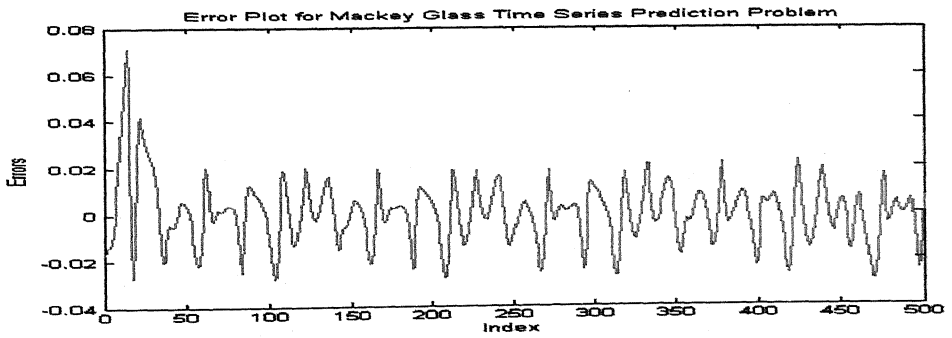

Fig 4.1.3.3 Errors at each pattern - Mackey-Glass Time Series Prediction – SA

4.2 Results by ALOPEX

4.2.1 Simulation Results for XOR Problem

The network architecture used for solving XOR is 2-2-1. After trying various combinations of N, T, δ, it is found that the combination of $N = 50, T = 1000$ and $\delta = 0.001$ works well. For the specified error tolerance of 1.0E-5, the error plot is shown in figure 4.2.1.1.

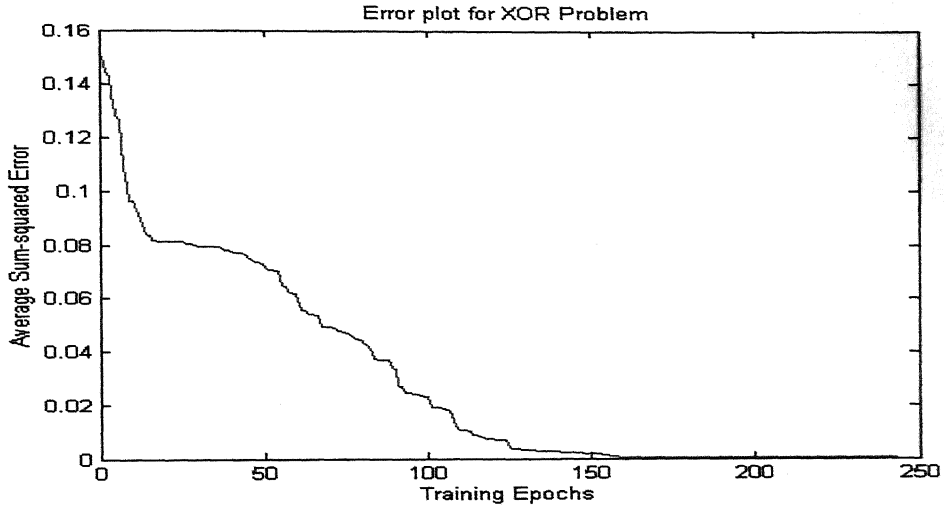

Fig 4.2.1.1 Error plot for XOR problem – ALOPEX

4.2.2 Simulation Results for 4-Parity Problem

In this work 4-bit parity problem has been considered. If the input pattern contains an even number of 1s, then its parity is 0 else it is 1. This is considered to be a difficult problem, because the patterns that are closer (using Euclidean distance) in the measure space, i.e. numbers that differ in only one bit require their answers to be different. The network architecture used to solve this problem is 4-4-4-1. For the specified error tolerance of 1.0 E-5, the results are shown in figure 4.2.2.1.

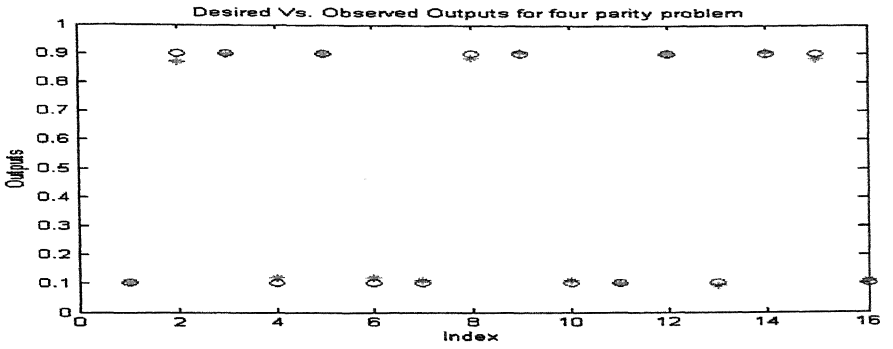

Fig 4.2.2.1 Desired and Observed Outputs for 4 Parity Problem – ALOPEX

4.2.3 Modeling a Three Input Nonlinear Function (TINF)

The training data for this problem are obtained from

$$output = \left(1 + x^{0.5} + y^{-1} + z^{-1.5}\right)^2$$

The training data and 125 checking data have been generated uniformly from the input ranges [1,6]*[1,6]*[1,6] and [1.5,5.5]* [1.5,5.5]* [1.5,5.5], respectively. The training data is used to train the network. Network architecture used to approximate this function is 3-3-1. The activation function used is unipolar sigmoid function. It has been observed that the value of δ should be very small. The values used for solving this problem are $N = 100, T = 1000$ and $\delta = 0.0001$. Corresponding plots are shown in figure 4.2.3.1 & 4.2.3.2.

The results of ALOPEX on XOR, 4-Parity, Sin(x)*Sin(y), TINF and Mackey Glass Time Series Prediction, D=4 (Mc-Glass4) problems are tabulated in Table 4.2.4.1.

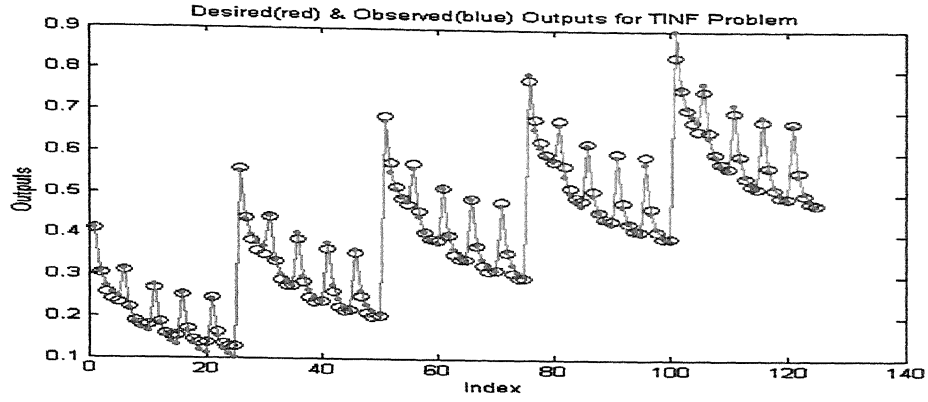

Fig 4.2.3.1 Desired and Observed Outputs - TINF – ALOPEX

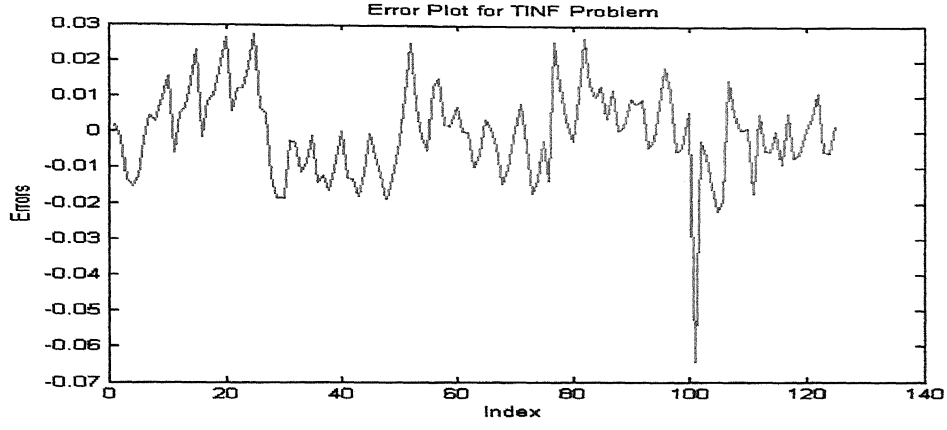

Fig 4.2.3.2 Error at each pattern - TINF – ALOPEX

	Architecture	Approximate Iterations	N	δ	T	Convergence
XOR	2-2-1	1500	50	0.01	1000	1E-5
4-Parity	4-4-4-1	3000	50	0.01	1000	1E-5
Sin(x)*Sin(y)	2-6-5-6-1	40000	100	0.0001	1000	1E-5
TINF	3-3-1	3000	50	0.01	1000	1E-5
Mc-Glass4	4-8-1	25000	100	0.0001	1000	1E-5

Table 4.4 ALOPEX Results

4.3 Results by Hybrid-SA Method

All the problems are trained for convergence 1.0 E-5. The results are tabulated as follows:

	Architecture	Approximate Iterations	ε'	G	Plots
XOR	2-2-1	220	50	0.01	4.3.1.1
4-Parity	4-4-4-1	3500	50	0.01	4.3.2.1, 4.3.2.2
Sin(x)*Sin(y)	2-6-5-6-1	8000	100	0.0001	4.3.4.1, 4.3.4.2
TINF	3-3-1	4000	50	0.01	-
Mc-Glass4	4-8-1	5000	100	0.0001	4.3.3.1, 4.3.3.2

Table 4.5 Results by Hybrid-SA

4.3.1 XOR Problem

The architecture used to solve XOR is 2-2-1. For the specified error tolerance of 1.0 E-5, the error plot is shown in fig 4.3.1.1.

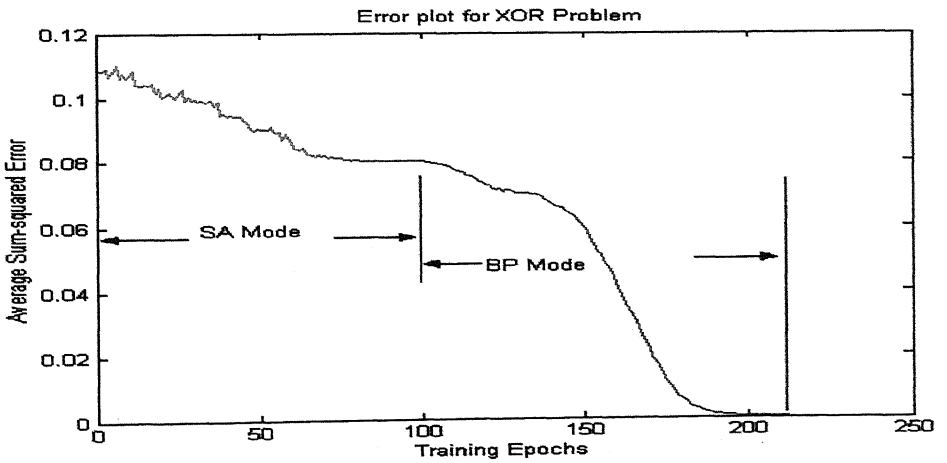

Fig 4.3.1.1 Error Plot for XOR Problem – SA-BP-SA

4.3.2 4-Parity Problem

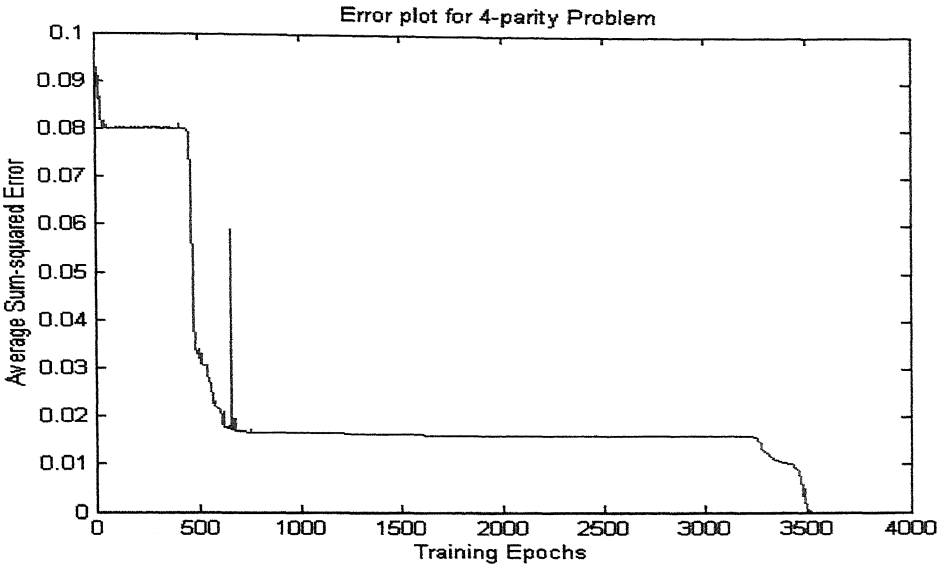

Fig 4.3.2.1 Error plot for 4-Parity Problem – Hybrid-SA

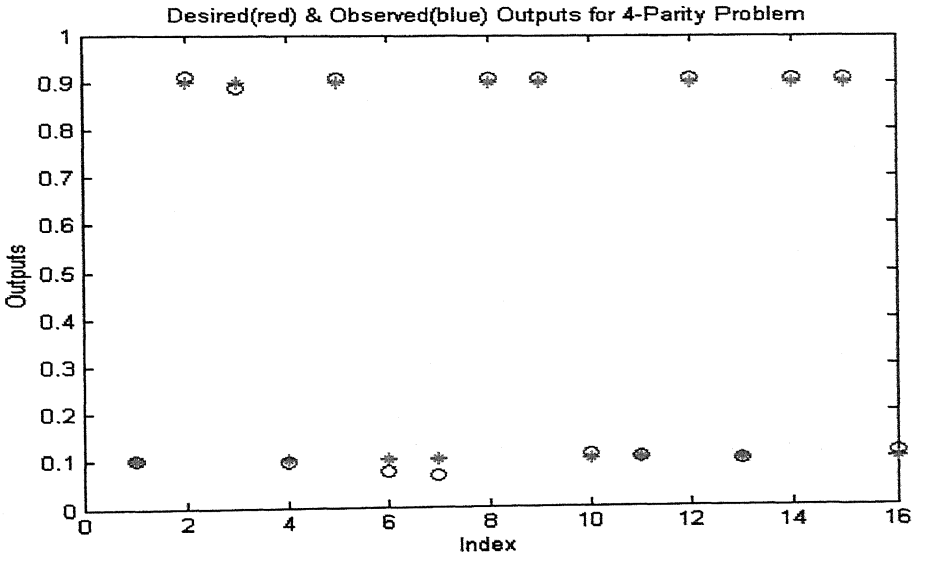

Fig 4.3.2.2 Desired & Observed Outputs for 4-Parity Problem

4.3.3 Mackey Glass Time Series Prediction

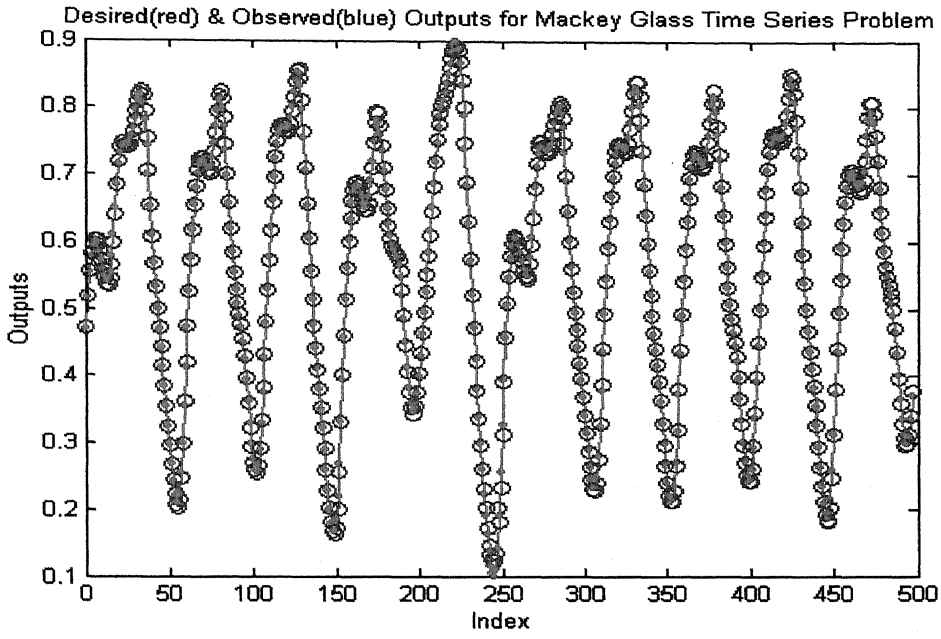

Fig 4.3.3.1 Desired & Observed Outputs for Mackey Glass Time Series Prediction -Hybird-SA

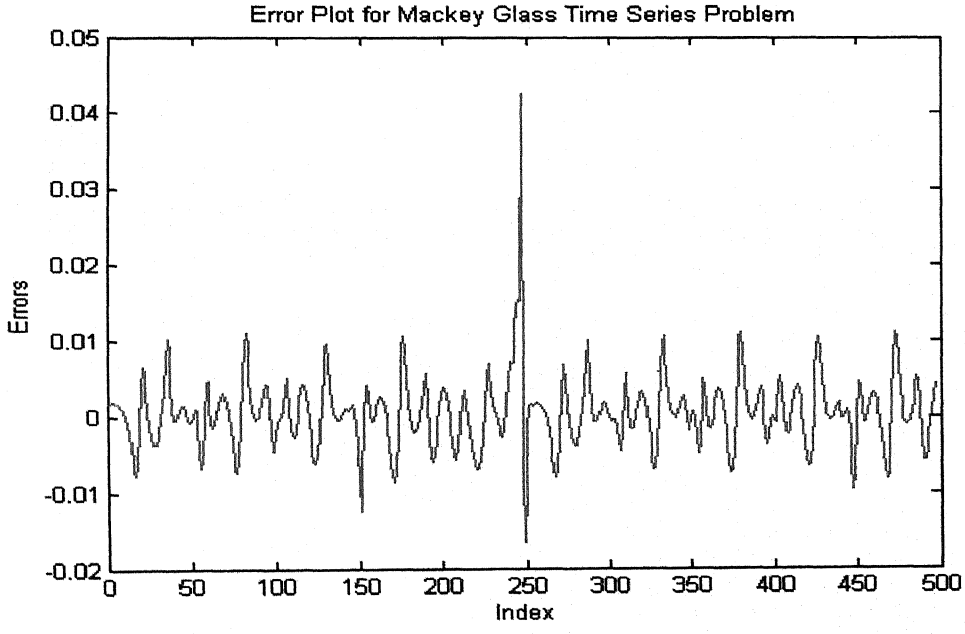

Fig 4.3.3.2 Error Plot for Mackey Glass Time Series Prediction - Hybrid-SA

4.3.4 Sin(x)*Sin(y) problem

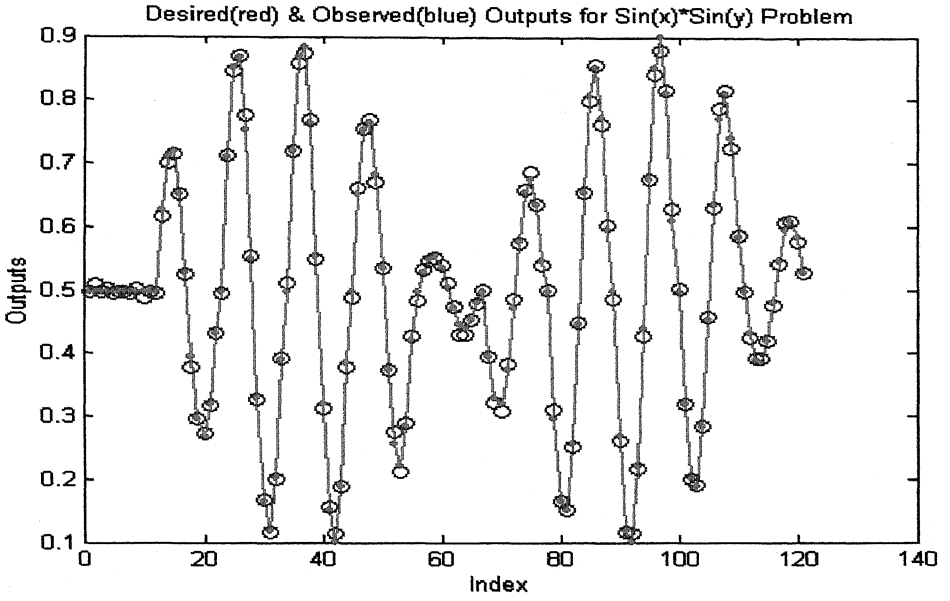

Fig 4.3.4.1 Desired and Observed Outputs for Sin(x)*Sin(y) problem – Hybrid-SA

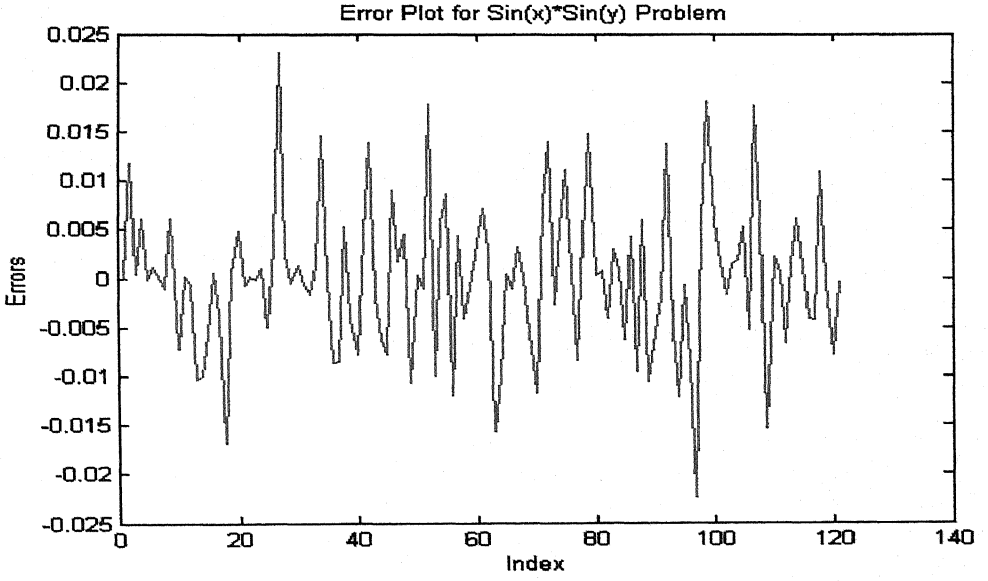

Fig 4.3.4.2 Error Plot for Sin(x)*Sin(y) problem

4.4 Results by SARPROP

To test the effectiveness of SARPROP, its performance is checked on the following problems. SARPROP uses the standard RPROP constant settings $\eta^+ = 1.2, \eta^- = 0.5, \Delta_{max} = 50, \Delta_{min} = 1*10^{-6}$. The value of T=0.01 is used for all benchmark problems.

4.4.1 Simulation Results for XOR problem

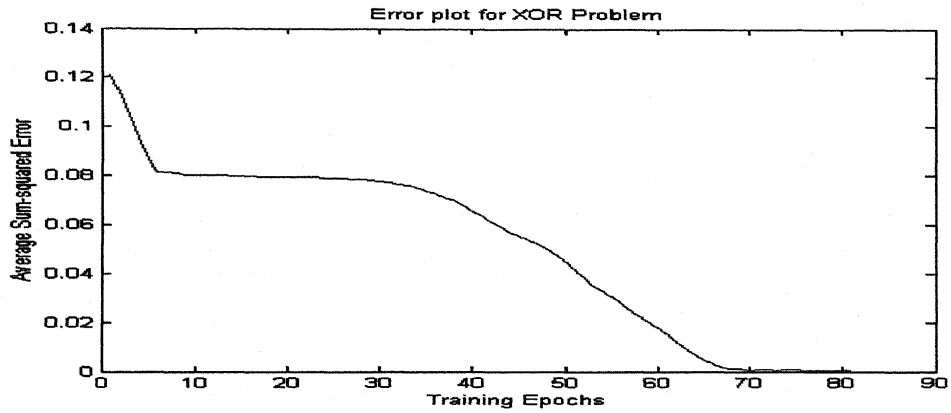

Fig 4.4.1.1 Error plot for XOR problem – SARPROP

4.4.2 Simulation Results for Sin(x)*sin(y) problem

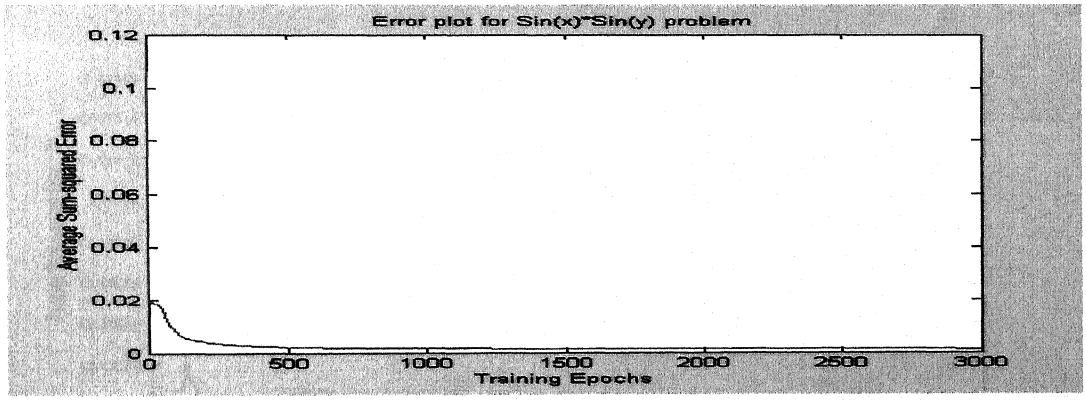

Fig 4.4.2.1 Error plot for Sin(x)*Sin(y) problem – SARPROP

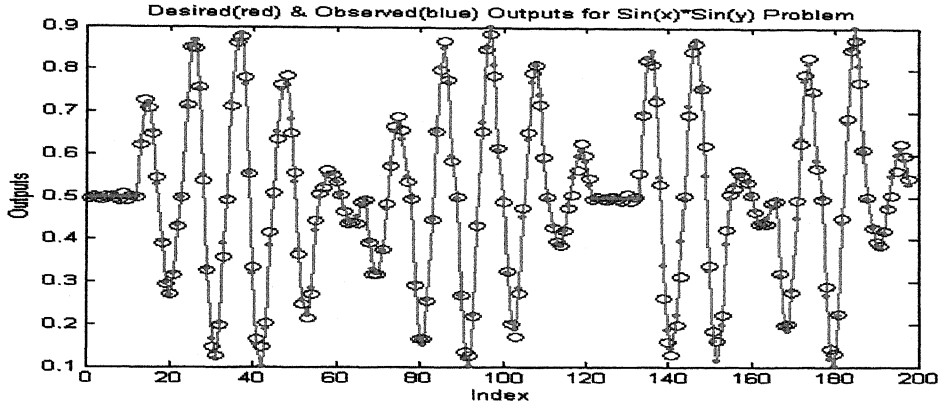

Fig 4.4.2.2 Desired and Observed Outputs for Sin(x)*Sin(y) problem

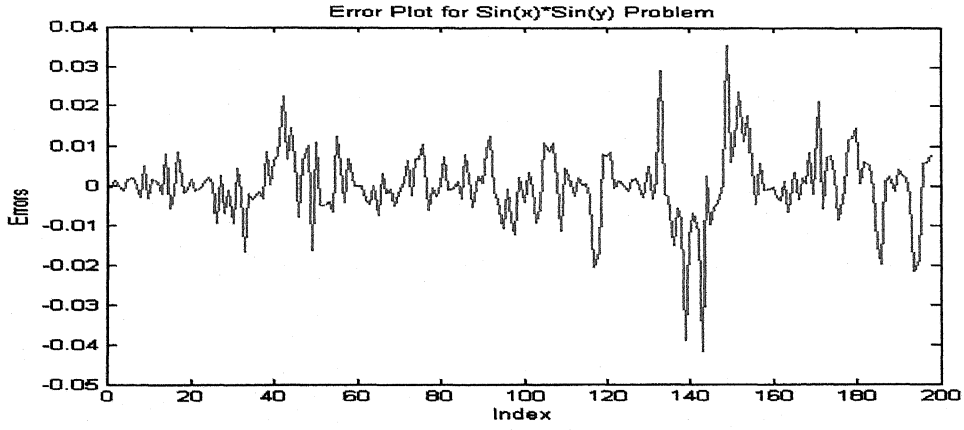

Fig 4.4.2.3 Errors at each pattern for Sin(x)*Sin(y) problem

4.4.3 Simulation results for TINF Problem

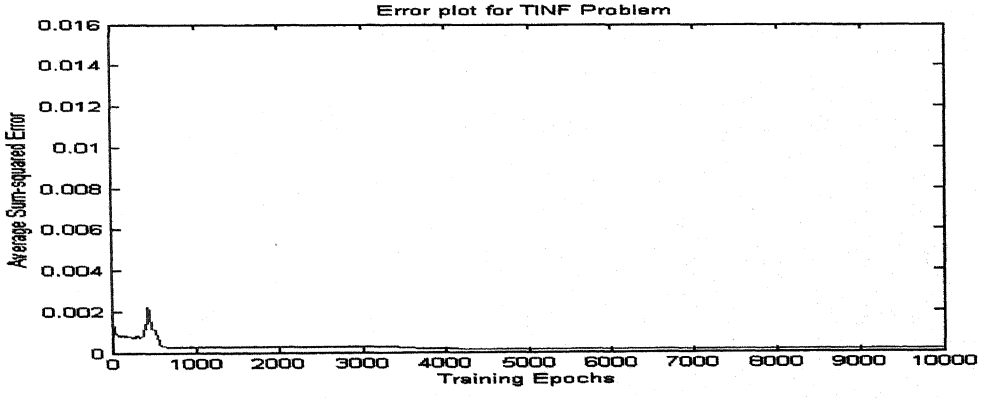

Fig 4.4.3.1 Error plot for TINF problem

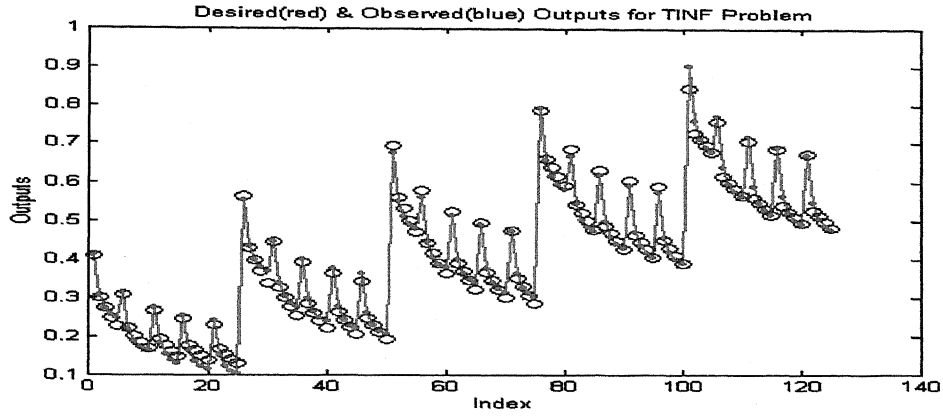

Fig 4.4.3.2 Desired and Observed Outputs for TINF problem

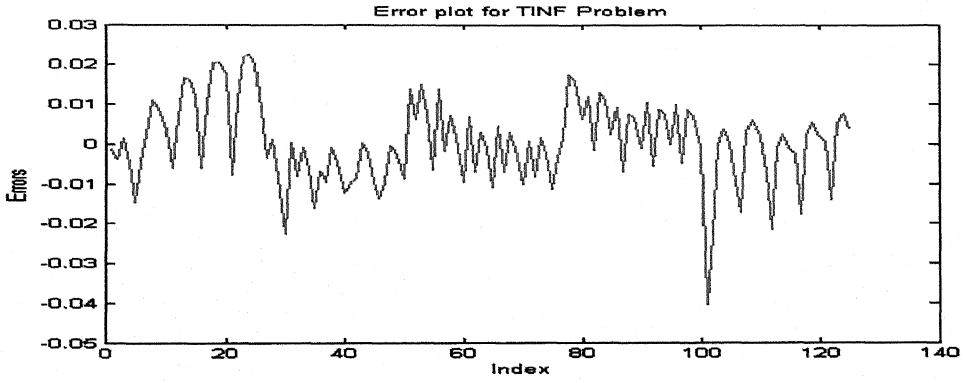

Fig 4.4.3.3 Errors at each pattern for TINF problem

4.4.4 Simulation results for Mackey-Glass Time series prediction Problem

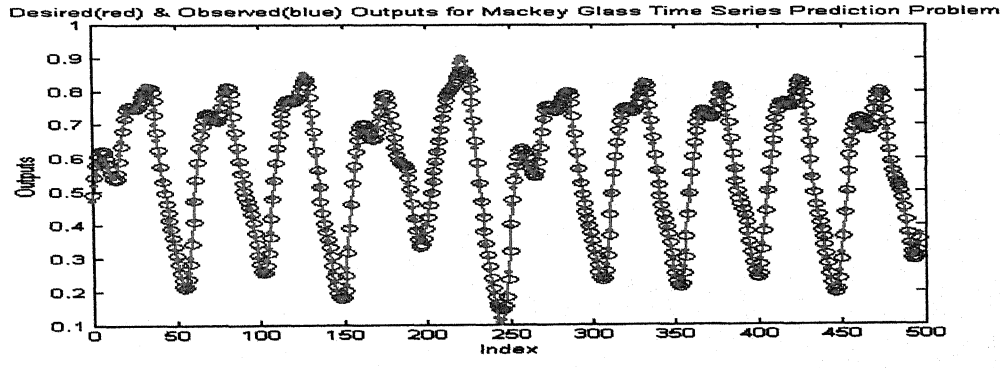

Fig 4.4.4.1 Desired and Observed Outputs for Mackey Glass Time Series Prediction problem

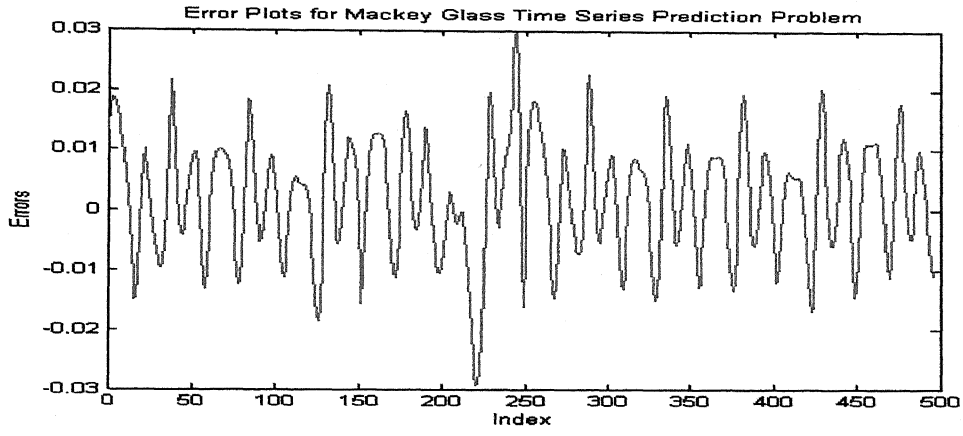

Fig 4.4.4.2 Errors at each pattern for Mackey Glass Time Series Prediction

4.5 Results by NOVEL

All the problems are trained for convergence 1.0 E-5. The results are tabulated as follows:

	Architecture	Approximate Iterations	μ_g, μ_t, ρ	Plots
XOR	2-2-1	480	1,1,1	4.6.1.1
4-Parity	4-4-4-1	360	1,1,1	4.6.2.1,4.6.2.2
Sin(x)*Sin(y)	2-6-5-6-1	10000	1,1,2	4.6.3.1,4.6.3.2,4.6.3.3
TINF	3-3-1	1350	1,1,1	-
Mc-Glass4	4-8-1	5000	1,1,2	4.6.4.1,4.6.4.2

Table 4.6 Results by NOVEL

4.5.1 Simulation Results for XOR Problem

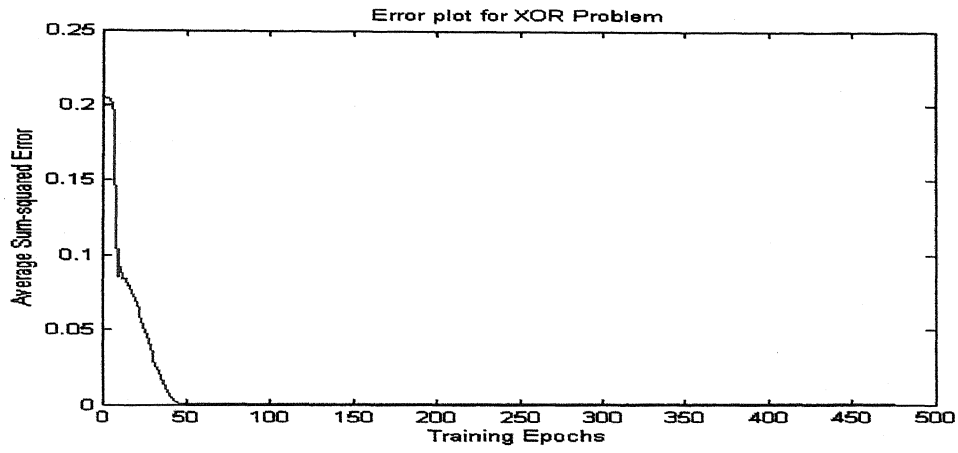

Fig 4.5.1.1 Error plot for XOR problem

4.5.2 Simulation Results for 4-Parity problem

The network architecture used is 4-4-4-1. The activation function used is unipolar sigmoid. The network took 363 epochs to get trained to the average error sum of 1 e-5.

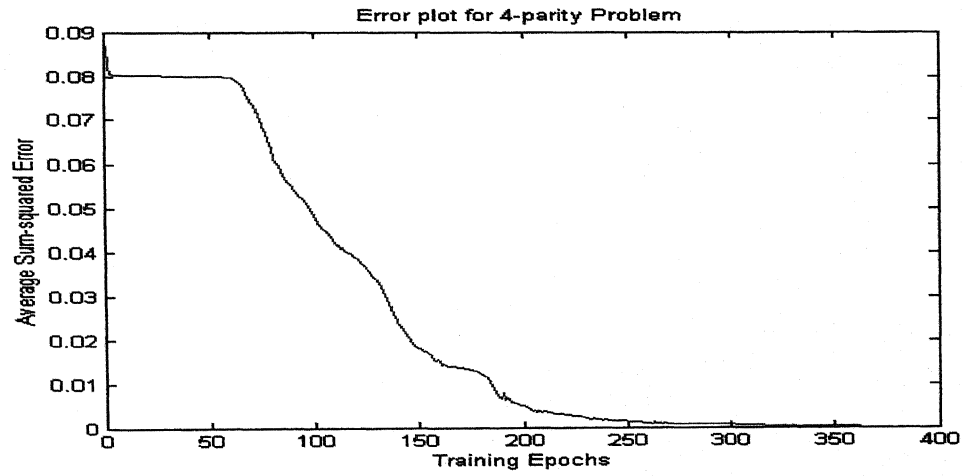

Fig 4.5.2.1 Error plot for Parity-4 problem

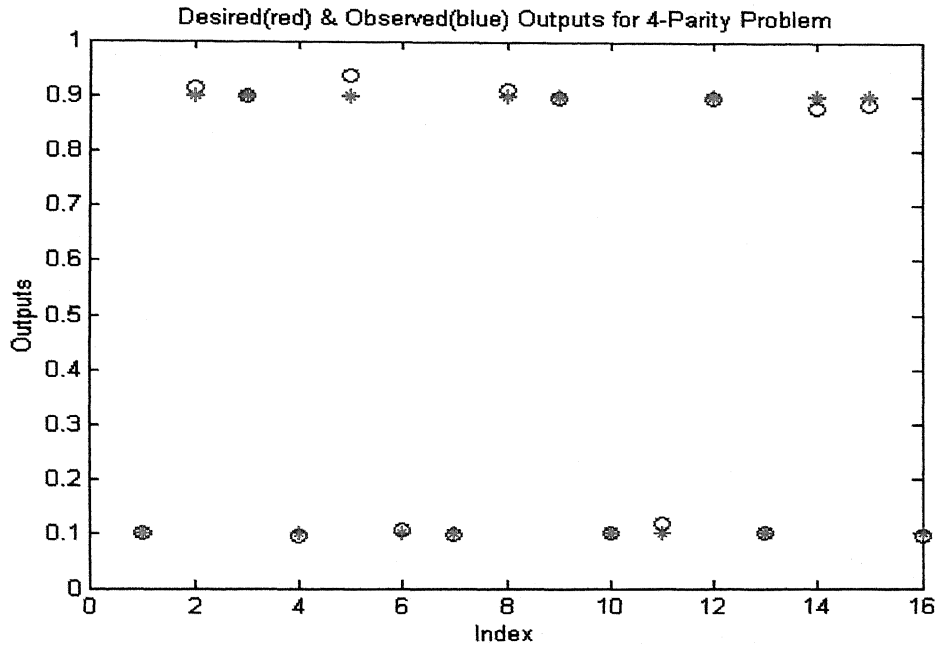

Fig 4.5.2.2 Desired and Observed Outputs for Parity-4 problem

4.5.3 Simulation Results for Sin(x)*Sin(y) problem

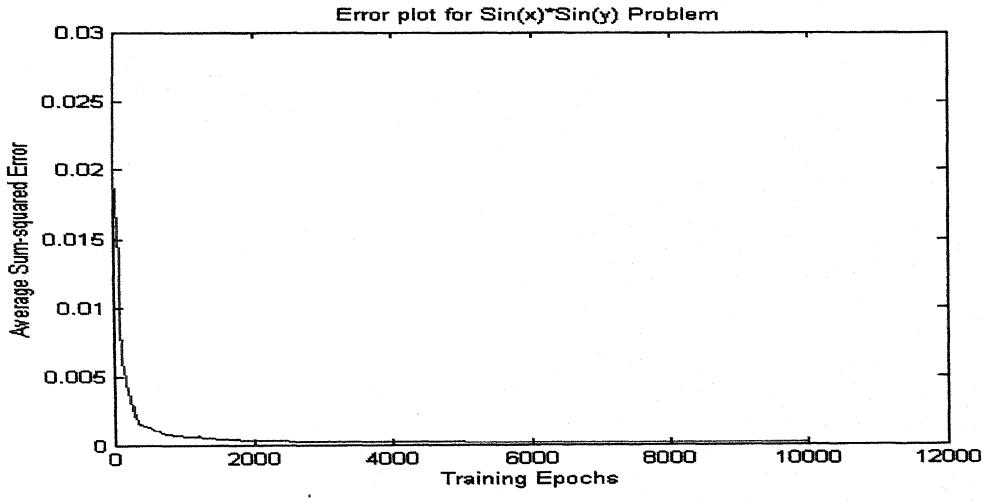

Fig 4.5.3.1 Error plot for Sin(x)*Sin(y) problem

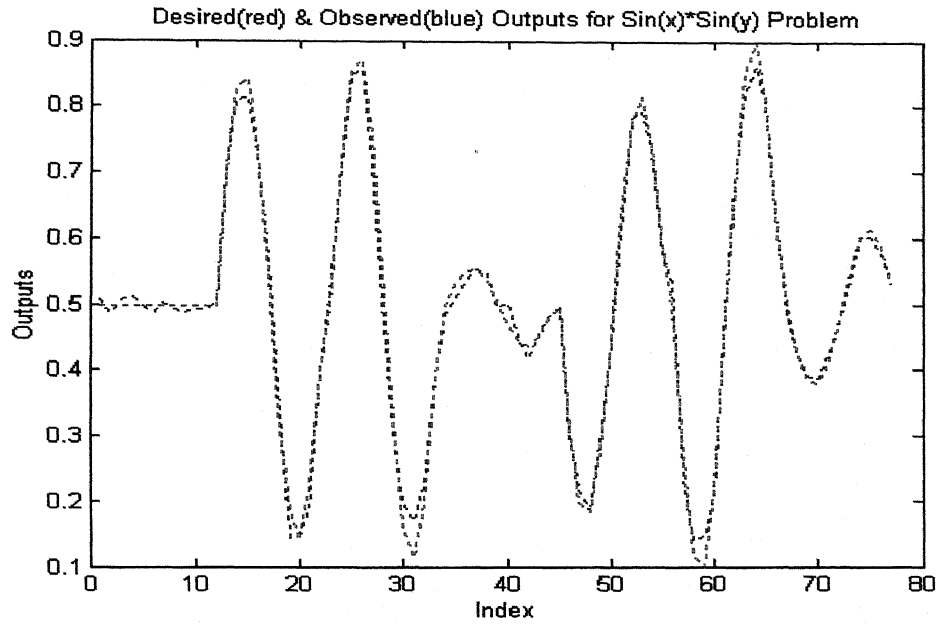

Fig 4.5.3.2 Desired and Observed Outputs for Sin(x)*Sin(y) problem

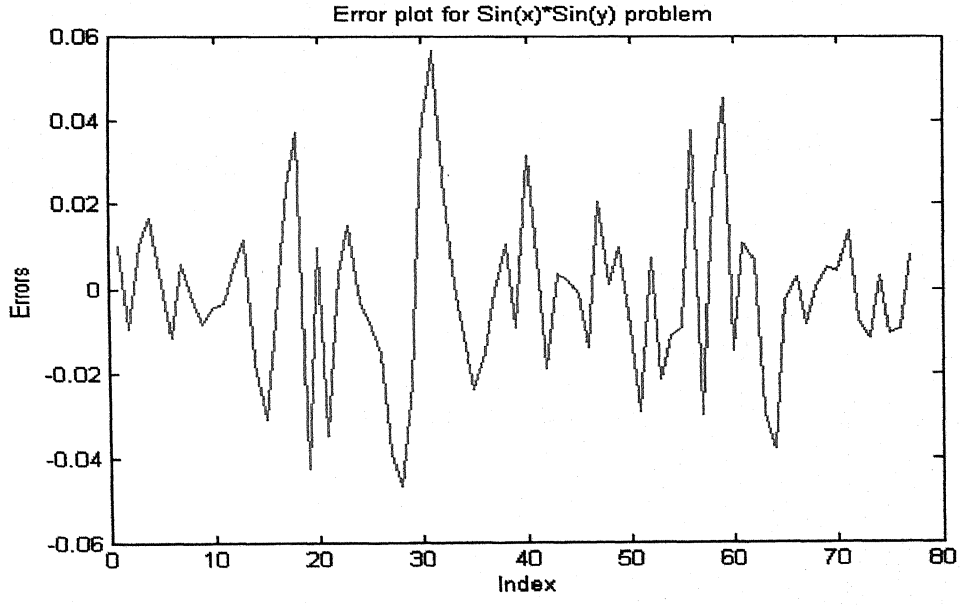

Fig 4.5.3.3 Error at each pattern for Sin(x)*Sin(y) problem

4.5.4 Simulation Results for Mackey Glass Time Series Prediction

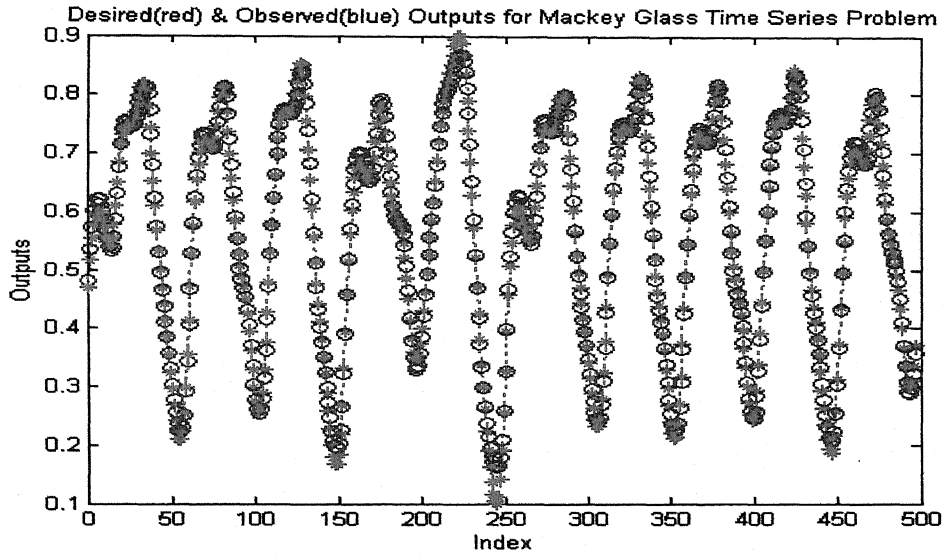

Fig 4.5.4.1 Desired & Observed Outputs for Mackey Glass Time Series Prediction by NOVEL

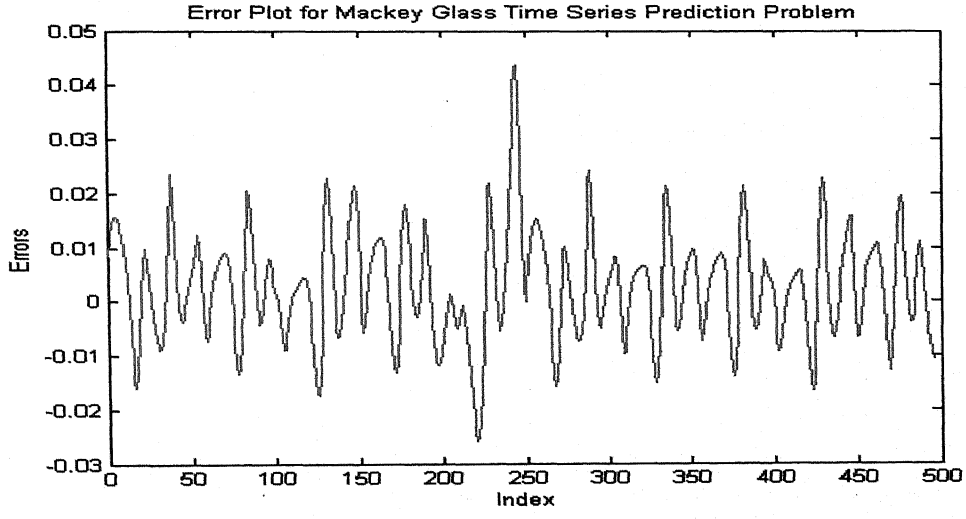

Fig. 4.5.4.2 Error at each pattern for Mackey-Glass Time Series Prediction by NOVEL

4.6 Results by Ray Guided Exploration Method (R-GEM)

4.6.1 Simulation Results for XOR Problem

Network Architecture used to solve XOR is 2-2-1. Figure 4.6.1.1 illustrates the advantage of R-GEM over R-PROP. R-GEM has taken 1650 (1825-175) iterations to confirm that the error surface is flat (Error sum is 0.04000000009582318). Once detected, it launches global search guided by emanating rays. As explained in the chapter 3, emanating rays has potential to get out of local minima. In this case, it has given a complete different weight space configuration whose error sum is 0.008517. Once again, the algorithm shifts to local method. Here RPROP as a local method is employed. It has taken only 225 (2051-1826) iterations to reach the convergence. The range of t is [-10,10].

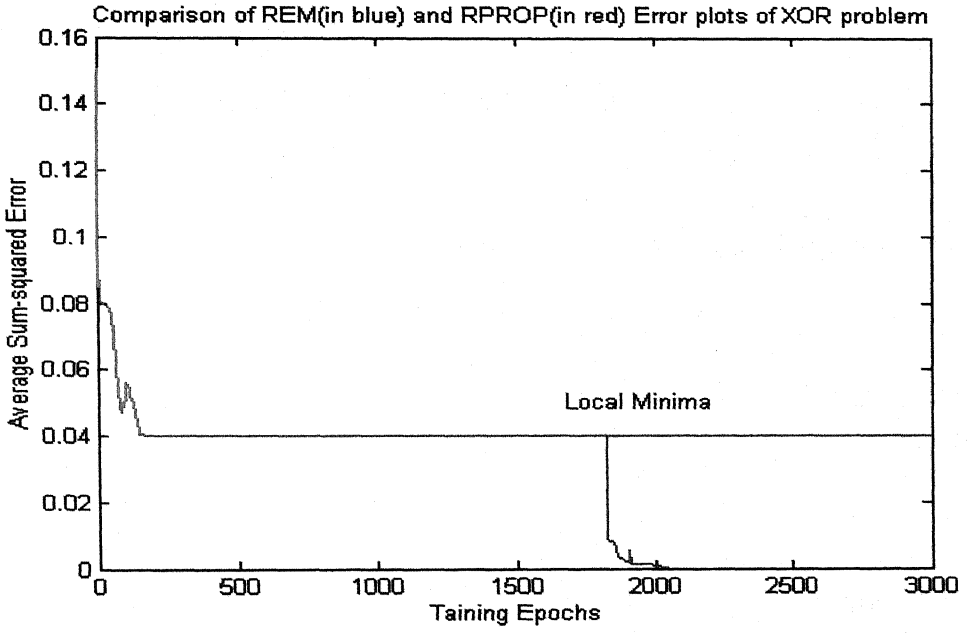

Fig 4.6.1.1 Comparison of R-GEM & RPROP Error plots for XOR problem

4.6.2 Simulation Results for Two Spiral Problem

This is a benchmarking classification problem. The problem consists of Np=198 (x,y) pairs of points that lie on interlocking spirals (Np/2 on each) separated in space by 180 degrees, that go around a common origin three times. Points on each spiral belongs to one class and the network has to associate points in the sample space with the correct class (0,1). This problem has been solved by R-GEM.

Network Architecture used to solve this problem is 2-20-20-20-1. Unipolar sigmoid as activation function is used for all layers. The range of t is [-20,20]. The procedure is outlined below:

1. First initial weights are stored.

2. It is trained with Back Propagation. Local minima resulted during error sum = 0.0099.

3. The local minima weights are stored.

4. Then by employing REM, using weights that resulted in local minima, we get ridden from local minima. The resultant weights are stored.

6. Then the network is reloaded with weights (obtained in step 4) and trained using back Propagation. This results in further refinement of error which is 1.99e-4.

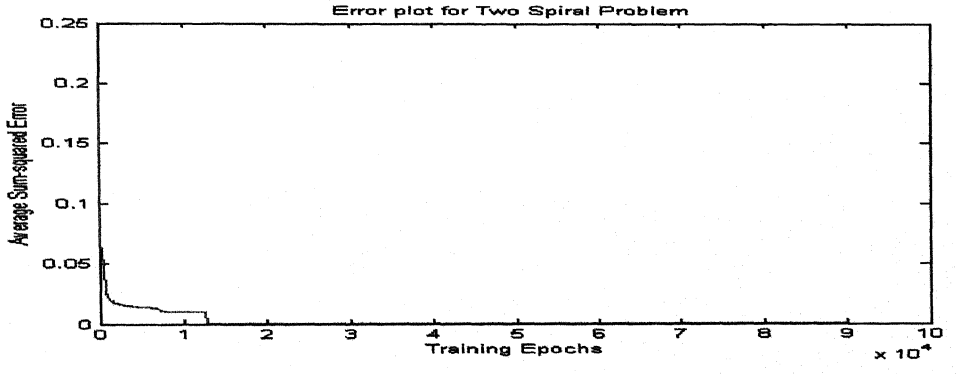

Fig. 4.6.2.1 Error plot for Two Spiral Problem – R-GEM

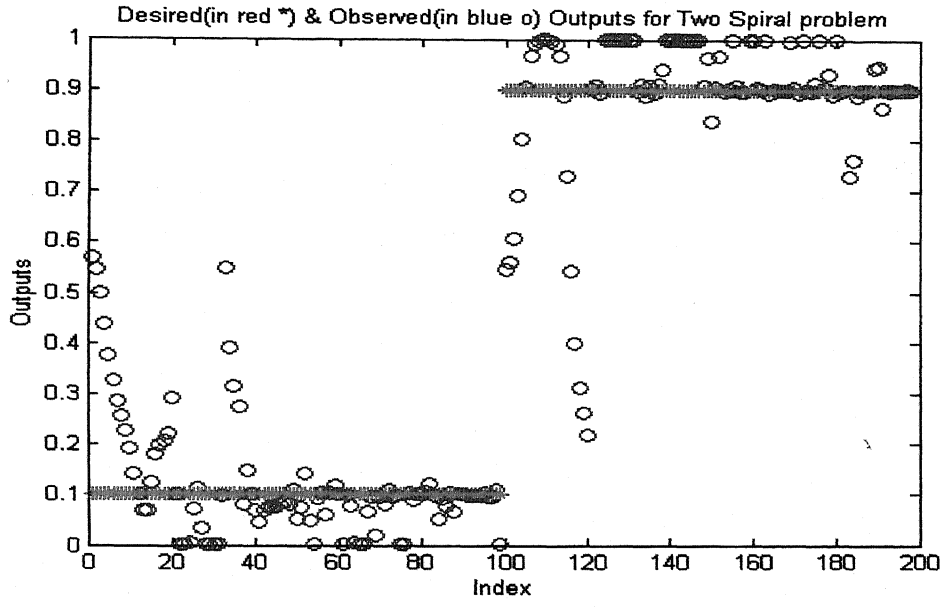

Fig. 4.6.2.2 Desired & Observed Outputs for Two Spiral Problem – Back Prop - local minima problem

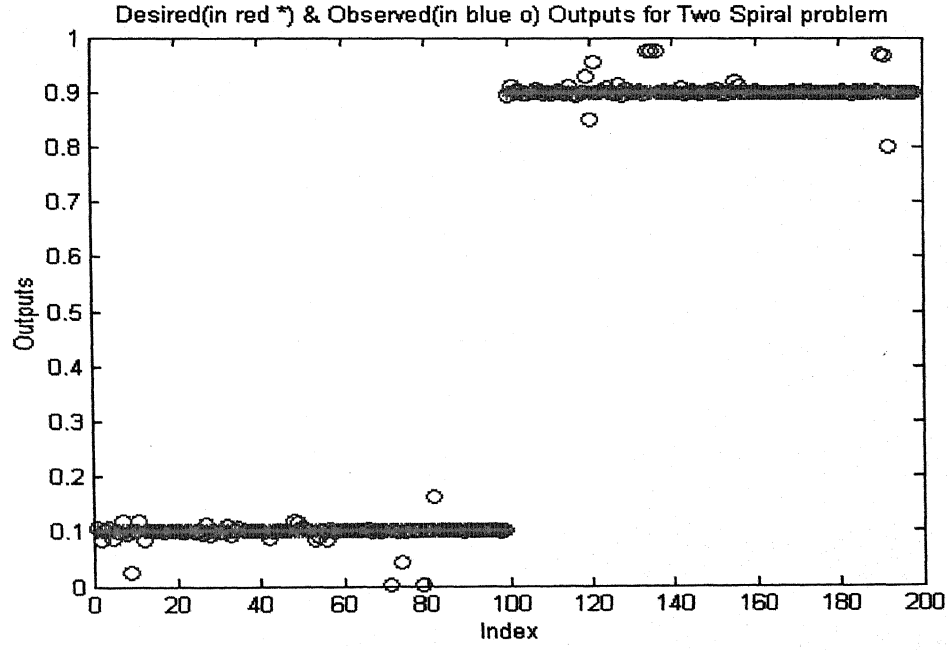

Fig. 4.6.2.3 Desired & Observed Outputs for Two Spiral Problem – R-GEM

4.6.3 Predicting Chaotic Dynamics by R-GEM

The architecture used to predict this Mackey Glass Time Series Problem is 4:8:1. The network is trained for a convergence of 1.0 E-5. Number of local minima occurred during training is 1.0E-5. Error at local minima: 3.8491364071779454E-4 occurred at 232nd iteration. Ray method generated six different configurations whose error sum is less than the local minima.

1: 3.8389114811329087E-4

2: 3.8381775271681477E-4

3: 3.8369154853762013E-4

4: 3.844312485308996E-4

5: 3.8464973271783403E-4

6: 3.7661994169421324E-4

And the best one is selected.

Error after ray method is 3.7661994169421324E-4. Improvement in reduction of error sum is 2.2%. The results are shown in figures 4.6.3.1, 4.6.3.2 and 4.6.3.3.

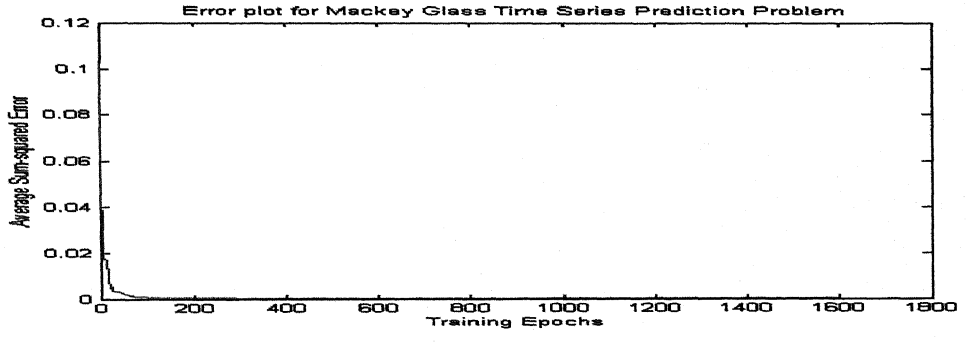

Fig 4.6.3.1 Error Plot for Mackey Glass Time Series Prediction – R-GEM

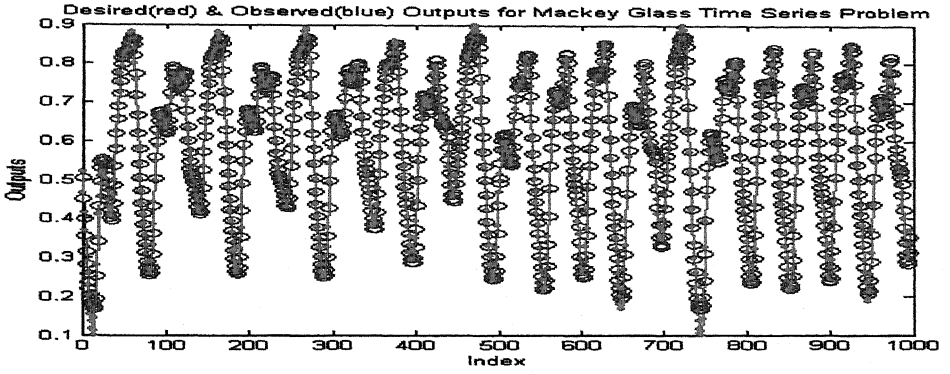

Fig 4.6.3.2 Desired & Observed Outputs for Mackey-Glass Time Series Prediction

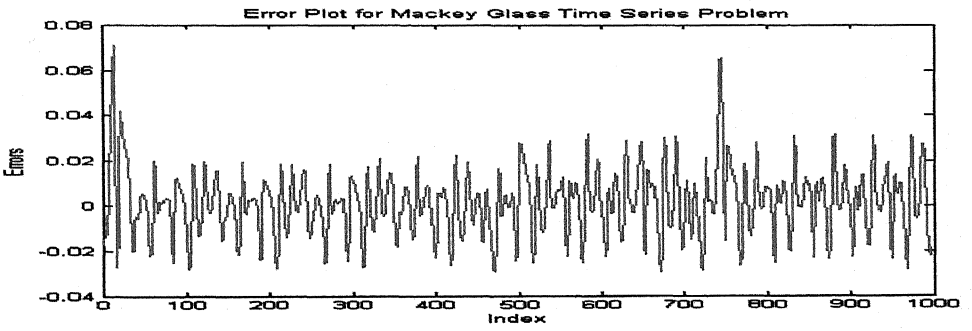

Fig 4.6.3.3 Errors at each pattern for Mackey-Glass Time Series Prediction

CONCLUSIONS

Summary

Learning in neural networks was viewed as a search mechanism for a minimum of a multi-dimensional error function. Local Minima Problem encountered in Back Propagation is addressed. Stochastic Methods like Simulated Annealing, ALOPEX were developed. New cooling schedules were explored. Since gradient information is not taken into account for updating weights, the computations are very simple. These methods eventually realize global minima even where the error surface is rugged and have many local minima. The developed algorithms are tested on several benchmark problems. Using stochastic methods, it is found that "Larger the network structure, longer is the training".

To overcome the disadvantages of stochastic methods, hybrid methods are developed. Hybrid-SA, which combines the advantages of back propagation and simulated annealing, is developed. Hybrid-SA suffers from poor weight initialization. To overcome the disadvantage of Hybrid-SA, SA-BP-SA is proposed. In SA-BP-SA, the initialized weights are tuned properly by Simulated Annealing. The heuristics for change of control from SA to Hybrid-SA are discussed.

A SA addition to the RPROP algorithm, SARPROP, has been developed. The success of the combination of noise and weight decay in increasing both the convergence speed has been demonstrated on a number of benchmark problems.

A hybrid global/local-minimization method, NOVEL is developed. This is a deterministic method which brings a search out of a local minimum. This relies on an external force, in the form of trace function, to pull the search out of a local minimum and employs local descents to locate local minima.

Ray-Guided Exploration Method, R-GEM is developed for finding good minima which uses emanating rays that traverse the error landscape. This method avoids many unnecessary computations in redetermining the already known regions as the selection of starting points comes from the information-bearing error curve.

Scope for further work

Some methods for trying different combination of various global optimization methods were discussed in this present work. Quite a lot of work should still be done in this area before understanding the weak and the strong points of different methods in real-life applications. Hybrid-global gradient methods are especially worth developing. Genetic Algorithms can be used in conjunction with Simulated Annealing called GA-SA. GA are used to speed up the SA in this case.

REFERENCES

[1] Ihor O. Bohachevsky, Mark E. Johnson, and Myron L. Stein, "Generalized Simulated Annealing for Function Approximation", Technometrics, Vol. 28, No. 3, pp. 209-217, 1986.

[2] David Abramson, Mohan Krishnamoorthy and Henry Dang, "Simulated Annealing Cooling Schedules for the School Timetabling Problem", Management Science, 37(1), 98-113, 1991.

[3] K.P. Unnivkrishnan, K.P. Venugopal, "Alopex: A Correlation-Based Learning Algorithm for Feedforward and Recurrent Neural Networks", Neural Computation, 1994.

[4] Norio Baba, Yoshio Mogami, Motokazu Kohzaki, Yashuiro Shiraishi and Yutaka Yoshida, " A Hybrid Algorithm for Finding the Global Minimum of Error Function of Neural Networks and Its Applications", Neural Networks, Vol. 7, No. 8, pp. 1253-1265, 1994.

[5] Nicholas K. Treadgold, Tam'as D. Gedeon, "Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm", IEEE Trans. Neural Networks, Vol. 9, pp. 662-668, 1998.

[6] Yi Shang and Benjamin W.Wah, "Global Optimization for Neural Network Training", IEEE Computer, Vol. 29, pp. 45-54, 1996.

[7] Ximin Zhang and Yan Qiu Chen, "Ray-guided global optimization method for training neural networks", Neurocomputing, Vol. 30, pp. 333-337, 2000.

[8] J.M. Zurada, Introduction to Artificial Neural Networks, Delhi, Jaico Publishing House, 1994.

[9] M.H. Hassoun, Fundamentals of Artificial Neural Networks, 1998.

[10] P.J.M. Van Laarhoven and E.H.L. Aarts Simulated Annealing: Theory and Applications, Kluwer Academic Publishers, 1987.

[11] Carl G. Looney, Pattern Recognition using Neural Networks, Oxford University Press, 1997.

[12] Dan W. Patterson, Artificial Neural Networks: Theory and Applications, Prentice Hall, 1995.

APPENDIX A

Statistical Mechanics is the central discipline of condensed matter physics, which deals with the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature. The starting point of statistical mechanics is an energy function $E(X)$ that measures the thermal energy of a physical system in a given state X, where X belongs to a set of possible states Σ. If the system's absolute temperature T is not zero $(i.e., T > 0)$, then the state X will vary in time, causing E to fluctuate. Being a physical system, the system will evolve its state in an average direction corresponding to that of decreasing energy E. This continues until no further decrease in the average of E is possible, which indicates that the system has reached thermal equilibrium. A fundamental result from physics is that at thermal equilibrium each of the possible states X occurs with the probability

$$P(X) = \frac{e^{-\frac{E(X)}{kT}}}{\sum_{X \in \Sigma} e^{-\frac{E(X)}{kT}}} \qquad \text{(A.1)}$$

where k is Boltzmann's constant, and the denominator is a constant that restricts $P(X)$ between zero and one. The above equation is referred as Boltzmann-Gibbs distribution.

Now define a set of transition probabilities $W(X \to X^1)$ from a state X into X^1. What is the condition on $W(X \to X^1)$ so that the system may reach and then remain in thermal equilibrium? A sufficient condition for maintaining equilibrium is that the average number of transitions from X to X^1 and from X^1 to X be equal:

$$P(X)W(X \to X^1) = P(X^1)W(X^1 \to X) \qquad \text{(A.2)}$$

or, by dividing by $W(X \to X^1)$ and using Equation (1),

$$\frac{W(X \to X^1)}{W(X^1 \to X)} = \frac{P(X^1)}{P(X)} = e^{-\frac{\Delta E}{kT}} \qquad \text{(A.3)}$$

where $\Delta E = E(X^1) - E(X)$. In simulating physical systems, a common choice for $W(X \to X^1)$ is the Metropolis algorithm, where

$$\begin{aligned} W(X \to X^1) &= 1 && \text{if } \Delta E < 0 \\ &= e^{-\frac{\Delta E}{kT}} && \text{otherwise} \end{aligned} \qquad \text{(A.4)}$$

This has the advantage of making more transitions to lower-energy states than those in Equation (3) and therefore reaches equilibrium more rapidly. Note that the transitions from low to high energy states are possible except when $T = 0$.

APPENDIX B

The RPROP algorithm is as follows:

$\forall i,j : \Delta_{ij}(t) = \Delta_0$

$\forall i,j : \frac{\partial E}{\partial w_{ij}(t-1)} = 0$

REPEAT

Compute Gradient $\frac{\partial E}{\partial w_{ij}(t)}$

For all weights and biases

IF $\frac{\partial E}{\partial w_{ij}(t-1)} * \frac{\partial E}{\partial w_{ij}(t)} > 0$

$\Delta_{ij}(t)$=**Minimum** $\left(\Delta_{ij}(t-1) * \eta^{+}, \Delta_{\max}\right)$

$\Delta w_{ij}(t)$= **-sign** $\left(\frac{\partial E}{\partial w_{ij}(t)}\right) * \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

$\frac{\partial E}{\partial w_{ij}(t-1)} = \frac{\partial E}{\partial w_{ij}(t)}$

ELSEIF $\frac{\partial E}{\partial w_{ij}(t-1)} * \frac{\partial E}{\partial w_{ij}(t)} < 0$

$\Delta_{ij}(t) = \Delta_{ij}(t-1) * \eta^{-}$

$\Delta_{ij}(t)$=**Maximum** $\left(\Delta_{ij}, \Delta_{\min}\right)$

$\frac{\partial E}{\partial w_{ij}(t-1)} = 0$

ELSEIF $\frac{\partial E}{\partial w_{ij}(t-1)} * \frac{\partial E}{\partial w_{ij}(t)} = 0$

$$\Delta w_{ij}(t) = \textbf{-sign}\left(\frac{\partial E}{\partial w_{ij}(t)}\right) * \Delta_{ij}(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

$$\frac{\partial E}{\partial w_{ij}(t-1)} = \frac{\partial E}{\partial w_{ij}(t)}$$

UNTIL CONVERGED

APPENDIX C

COMPARISON OF ALGORITHMS

The Complexity of an algorithm can be either the weight complexity, the time complexity indicated by the number of operations per pattern presented while testing or the number of iterations taken by the algorithm while training (specified error tolerance). The weight complexity varies from problem to problem, where as the time complexity for a pattern per neuron remains constant. Here the time complexity of all algorithms remains constant as the structure for all the algorithms are same. All the algorithms are tested on those initial weight configurations which Back Propagation failed to reach / realize convergence.

	XOR	SIN(X)*SIN(Y)	McGlass Time Series
NETWORK	2-2-1	2-6-5-6-1	4-8-1
SA	370	40000	10000
ALOPEX	1500	40000	25000
Hybrid-SA	1825+220	8000	5000
SARPROP	2500	3000	4000
NOVEL	1825+480	10000	5000
R-GEM	1825+225	10000	1600

Table: Comparison of different Algorithms based on number of training iterations

APPENDIX D

Mackey Glass Time Series Prediction

The Mackey Glass equation is given by:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$$

where $\tau = 1.7, x(0) = 1.2, x(t) = 0$ for $t < 0$. A total of 1000 points have been generated with index t running from t=0 to t=1000. The most surprising finding was that the prediction could be done for the subject chaotic series with a network having only one layer. A total of 500 points has been used for training the network and the rest have been used for testing. The activation function is the sigmoid through out.

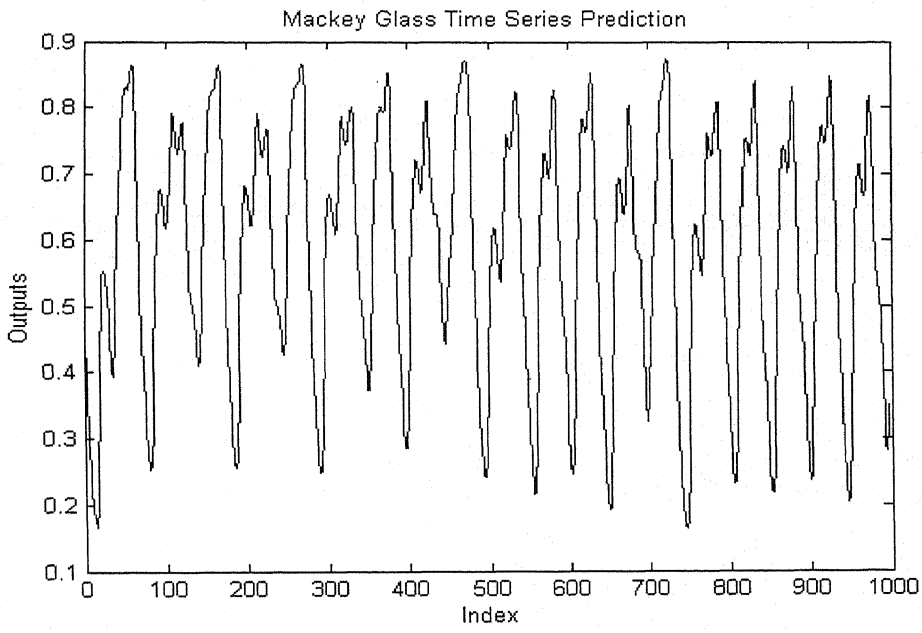

Figure: Mackey Glass Time Series Prediction (D=4)